

CU  
②  
**NAVAL POSTGRADUATE SCHOOL**  
**Monterey, California**

**AD-A246 208**



**DTIC**  
**ELECTE**  
**FEB 21 1992**  
**S B D**

**THESIS**

**AN EMPIRICAL APPROACH  
TO ANALYSIS OF SIMILARITIES  
BETWEEN SOFTWARE FAILURE REGIONS**

by

Lelon Levoy Ginn

September 1991

Thesis Advisor:

Timothy J. Shimeall

Approved for public release; distribution is unlimited.

**92-03921**



92 11 052

## REPORT DOCUMENTATION PAGE

1a. REPORT SECURITY CLASSIFICATION <b>UNCLASSIFIED</b>		1b. RESTRICTIVE MARKINGS	
2a. SECURITY CLASSIFICATION AUTHORITY		3. DISTRIBUTION/AVAILABILITY OF REPORT Approved for public release; distribution is unlimited	
2b. DECLASSIFICATION/DOWNGRADING SCHEDULE			
4. PERFORMING ORGANIZATION REPORT NUMBER(S)		5. MONITORING ORGANIZATION REPORT NUMBER(S)	
6a. NAME OF PERFORMING ORGANIZATION Computer Technology Curriculum Naval Postgraduate School	6b. OFFICE SYMBOL (if applicable) 37	7a. NAME OF MONITORING ORGANIZATION Naval Postgraduate School	
6c. ADDRESS (City, State, and ZIP Code) Monterey, CA 93943-5000		7b. ADDRESS (City, State, and ZIP Code) Monterey, CA 93943-5000	
8a. NAME OF FUNDING/SPONSORING ORGANIZATION	8b. OFFICE SYMBOL (if applicable)	9. PROCUREMENT INSTRUMENT IDENTIFICATION NUMBER	
8c. ADDRESS (City, State, and ZIP Code)		10. SOURCE OF FUNDING NUMBERS	
		PROGRAM ELEMENT NO.	PROJECT NO.
		TASK NO.	WORK UNIT ACCESSION NO.
11. TITLE (Include Security Classification) <b>AN EMPIRICAL APPROACH TO ANALYSIS OF SIMILARITIES BETWEEN SOFTWARE FAILURE REGIONS (U)</b>			
12. PERSONAL AUTHOR(S) Ginn, Lelon L.			
13a. TYPE OF REPORT Master's Thesis	13b. TIME COVERED FROM <b>Oct 90</b> TO <b>Sept 91</b>	14. DATE OF REPORT (Year, Month, Day) <b>September 1991</b>	15. PAGE COUNT <b>154</b>
16. SUPPLEMENTARY NOTATION The views expressed in this thesis are those of the author and do not reflect the official policy or position of the Department of Defense or the U. S. Government.			
17. COSATI CODES		18. SUBJECT TERMS (Continue on reverse if necessary and identify by block number)	
FIELD	GROUP	SUB-GROUP	
		Failure Regions, Software Testing, Cluster Analysis, Software Experiments, Software Faults, Software Fault Detection	
19. ABSTRACT (Continue on reverse if necessary and identify by block number) Previous authors have postulated that faults are related to each other and testers have tried to exploit the effect. However, the evidence and applications have been largely anecdotal. This thesis uses an analytical derivation of software failure regions to develop a quantitative metric of the relationship of one fault to another. This metric is then applied in an empirical study of a population of failure regions derived from faults used in a previous experiment. The failure regions were analyzed for clustering behavior using graph theory techniques. The goal of this study is to be able to use information about known faults in a program as a means of finding other faults in the same program. This study provides strong evidence that failure regions have a tendency to form clusters. Further, two specific characteristics of failure regions that lead to cluster formation are identified: shared bounding conditions (the Identical dimension) and shared variables that appear in different contexts (the Coincidental dimension). The nature of the clusters formed by these two dimensions are markedly different. The Identical dimension clusters are small, isolated, and strongly connected. The Coincidental dimension clusters are larger and more loosely connected. Software testing implications of failure region clustering behavior are discussed.			
20. DISTRIBUTION/AVAILABILITY OF ABSTRACT <input checked="" type="checkbox"/> UNCLASSIFIED/UNLIMITED <input type="checkbox"/> SAME AS RPT. <input type="checkbox"/> DTIC USERS		21. ABSTRACT SECURITY CLASSIFICATION <b>UNCLASSIFIED</b>	
22a. NAME OF RESPONSIBLE INDIVIDUAL Timothy J. Shimeall		22b. TELEPHONE (Include Area Code) (408) 646-2509	22c. OFFICE SYMBOL CS/ Sm

Approved for public release; distribution is unlimited

**AN EMPIRICAL APPROACH  
TO ANALYSIS OF SIMILARITIES  
BETWEEN SOFTWARE FAILURE REGIONS**

by  
Lelon Levoy Ginn  
Lieutenant, United States Navy  
Bachelor of Science, Lubbock Christian College, 1981

Submitted in partial fulfillment of the  
requirements for the degree of

**MASTER OF COMPUTER SCIENCE**

from the

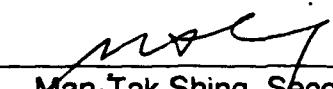
**NAVAL POSTGRADUATE SCHOOL**  
September 1991

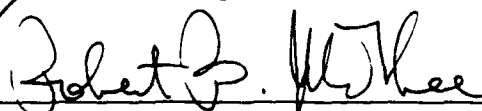
Author:

  
Lelon Levoy Ginn

Approved By:

  
Timothy J. Shimeall, Thesis Advisor

  
Man-Tak Shing, Second Reader

  
Robert B. McGhee, Chairman,  
Department of Computer Science

## ABSTRACT

Previous authors have postulated that faults are related to each other and testers have tried to exploit the effect. However, the evidence and applications have been largely anecdotal. This thesis uses an analytical derivation of software failure regions to develop a quantitative metric of the relationship of one fault to another. This metric is then applied in an empirical study of a population of failure regions derived from faults used in a previous experiment. The failure regions were analyzed for clustering behavior using graph theory techniques. The goal of this study is to be able to use information about known faults in a program as a means of finding other faults in the same program. This study provides strong evidence that failure regions have a tendency to form clusters. Further, two specific characteristics of failure regions that lead to cluster formation are identified: shared bounding conditions (the Identical dimension) and shared variables that appear in different contexts (the Coincidental dimension). The nature of the clusters formed by these two dimensions are markedly different. The Identical dimension clusters are small, isolated, and strongly connected. The Coincidental dimension clusters are larger and more loosely connected. Software testing implications of failure region clustering behavior are discussed.



iii

Accession For	
NTIS GRA&I	<input checked="checked" type="checkbox"/>
DTIC TAB	<input type="checkbox"/>
Unannounced	<input type="checkbox"/>
Justification	
By	
Distribution/	
Availability Codes	
Dist	Avail and/or Special
A-1	

## TABLE OF CONTENTS

I.	INTRODUCTION .....	1
A.	MOTIVATION FOR THIS STUDY .....	1
B.	BACKGROUND .....	2
C.	HYPOTHESIS .....	3
D.	DESCRIPTION OF THE EXPERIMENT .....	4
E.	OVERVIEW OF THE THESIS .....	4
II.	BACKGROUND AND RELATED WORK .....	5
A.	SOFTWARE TESTING .....	5
1.	Faults and Failures .....	5
2.	Software Fault Elimination .....	5
a.	Two Different Approaches to Software Testing .....	6
b.	A Theory of Test Data Selection .....	7
c.	Mutation Testing .....	8
d.	Partition Testing .....	9
3.	Failure Regions .....	10
B.	CLUSTER ANALYSIS .....	11
1.	Definitions .....	11
2.	The Basis for Using a Cluster Analysis Approach .....	12
3.	Cluster Analysis Techniques .....	12
a.	A Graph Theory Approach .....	12
b.	A Traditional Clustering Approach .....	13
C.	CONCLUSION .....	14
III.	EXPERIMENTAL DESIGN .....	16
A.	INTRODUCTION .....	16
B.	DESCRIPTION OF THE DATA .....	16
C.	DATA REDUCTION .....	17
1.	The Use of Predicates .....	18
2.	Analysis of Dimension Participation in Failure Region Bounds .....	19
3.	The Failure Region-Dimension Incidence Matrices .....	20
4.	Failure Region Graphs .....	20
5.	Determination of Edge Weights .....	21

a.	Separate Analysis of Identical and Coincidental Data . . . .	21
b.	Selection of the Weighting Coefficient . . . . .	22
D.	CLUSTER ANALYSIS . . . . .	24
1.	Clustering Method . . . . .	24
2.	Hierarchical Vs. Partitioned Clusters . . . . .	25
E.	CONCLUSION . . . . .	25
IV.	EMPIRICAL RESULTS . . . . .	27
A.	INTRODUCTION . . . . .	27
B.	DATA PRESENTATION . . . . .	27
C.	DATA ANALYSIS . . . . .	29
1.	Notable Characteristics . . . . .	29
2.	Data Validation . . . . .	30
3.	Cluster Formation . . . . .	34
D.	DATA INTERPRETATION . . . . .	37
E.	CONCLUSION . . . . .	37
V.	CONCLUSIONS AND SUGGESTIONS FOR FURTHER RESEARCH . . .	39
A.	CONCLUSIONS . . . . .	39
B.	RELATIONSHIP OF RESULTS TO PREVIOUS WORK . . . . .	40
C.	SUGGESTIONS FOR FURTHER RESEARCH . . . . .	41
1.	Experimental Method . . . . .	41
2.	Related Questions . . . . .	41
D.	APPLICATIONS BEYOND TESTING . . . . .	42
APPENDIX A	GRAPH THEORY DEFINITIONS . . . . .	44
APPENDIX B	VERSION 1 FAILURE REGIONS . . . . .	45
APPENDIX C	FAILURE REGION-VARIABLE INCIDENCE MATRICES . . . .	65
APPENDIX D	FAILURE REGION-VARIABLE INCIDENCE MATRIX ANALYSIS CODE . . . . .	84
APPENDIX E	THRESHOLD GRAPH EDGE LISTINGS . . . . .	90
APPENDIX F	HISTOGRAMS . . . . .	126
	LIST OF REFERENCES . . . . .	145
	INITIAL DISTRIBUTION LIST . . . . .	147

## **ACKNOWLEDGEMENTS**

The UNIX|STAT Data Analysis Programs by Gary Perlman were used to perform the statistical analysis of the experimental data. The thesis was prepared using Framemaker 3.0 by Frame Technology Corporation.

I would like to thank Commander Rachel Griffin and Dr. Kim Hefner for their comments and advice in the early analysis of the problem. I would also like to thank Dr. Man-Tak Shing for his willingness to substitute as my second reader on such short notice.

Last, and certainly most, I would like to thank Virginia, Byron, and Susan for their support while I was writing. They were more tolerant of me than I deserved.

## **I. INTRODUCTION**

### **A. MOTIVATION FOR THIS STUDY**

On July 10, 1991 an article in the Wall Street Journal blamed major telephone outages on a software failure. Three incorrectly set flag bits resulted in the omission of congestion control algorithms. DSC Communications Corp., the manufacturer of the faulty signaling system, reported that

... Pacific Bell ... had requested software changes involving perhaps three or four lines of code. Engineers decided that because the change was minor, the massive program didn't need to undergo the rigorous 13-week test that most software is put through before it is shipped to customers. (Wall Street Journal, 1991)

Engineers were also unable to explain why the problems didn't appear until several weeks after installation, and then only in two of the five Bell companies where the revised software was installed.

The decision by the engineers to forego testing because the changes were "minor" indicates a misunderstanding about how various parts of the program interact with each other. Their confusion about the delayed and selective appearances of the faults points to a lack of understanding about the conditions that had to be met for the fault to produce a failure. These problems clearly show the need for a method of projecting how changes will affect the performance of a program and how to deal with the conditions that cause those effects.

As general-purpose computing systems perform more and more sophisticated functions, the software necessarily becomes larger and more complex. As software size and complexity grow linearly, software testing and debugging become exponentially harder. In fact, testing consumes as much as half of the budget for the development of most major software systems, while error correction and specification revision account for up to 90% of software life-cycle costs after the software has been marketed (Alberts, 1976).



Unfortunately, extensive software testing is frequently necessary in spite of its expense. Many computer applications require fault-free, or at least fault-tolerant, operation. Examples include aircraft control and medical systems. In applications such as these, computer failure may result in a disaster, such as the loss of life or capital equipment. Even for systems that are less critical, such as the telephone example described above, failure can cause a significant loss of time, money, or productivity.

The need for reliable computers can only be expected to grow. This implies the need for reliable software since software failures are responsible for the majority of failures in computing systems that have fault tolerant hardware. Careful specification, design, and testing are the keys to producing reliable software. This thesis deals with the area of software testing.

The following sections briefly outline the background for this study, the hypothesis of the study, and a description of the experiment that was used to test the hypothesis.

## **B. BACKGROUND**

The ANSI/IEEE standard definition of a fault is an accidental condition that may cause a program to fail. Failure means that a program does not perform its required function. This may mean that the program does not execute or that it executes and produces an error. An error is a discrepancy between a computed value or condition and the true, specified or theoretically correct value or condition. (Glossary, 1983)

A subset of the program domain (i.e., input space) is associated with every fault in a program. Sets of bounds delimit this subset, one set corresponding to each variable in the domain. These bounds identify the values of the program variables that will result in program failure due to that specific fault. Every variable must be within its specified bounds before that fault will produce a failure. Ammann and Knight called the subset of the domain associated with a fault its failure region (Ammann and Knight, 1988). They determined failure regions empirically, by repetitive probing, rather than analytically.

Bolchoz described three conditions that are required for a fault to produce a failure. First, all the conditions for the fault to be executed must be met. Second, the fault must be executed in a way that produces an error. Finally, the error must be propagated to a final result without being masked by subsequent processing. The failure region of a fault is the subset of the program domain that allows the fault to satisfy all of these conditions simultaneously (Bolchoz, 1990).

### **C. HYPOTHESIS**

Bolchoz's study considered how to identify the failure regions of isolated faults. He did not consider relationships between faults. Elements of his analysis, however, suggested that failure regions may exhibit a relationship that links faults to each other. Failure regions are derived directly from their associated faults. Therefore, a relationship between failure regions would imply a relationship between their associated faults. If such a relationship exists, then the failure region of a known fault may be useful in deriving information about other failure regions. This information may, in turn, may lead to the discovery of other faults.

The primary goal of this research was to develop a technique for empirically examining failure regions to determine what relationships exist between failure regions. A secondary goal was to characterize the relationships. The hope is that these relationships may be useful in fault-detection applications.

Some difficulties arose during the development of the analysis technique. The first was that there was no statistical information about the behavior of failure regions. Which features of failure regions should be used in characterizing their behavior? What type of distribution does their behavior exhibit?

A second problem was the dimensionality of failure regions. A failure region has a separate dimension associated with each program variable. Failure regions for practical software can easily have several hundred dimensions. All of these dimensions are not necessarily orthogonal.

A third difficulty was how to quantify similarities between failure regions. Ideas such as Euclidean distance have no meaning because of the heterogeneity of the failure regions. How can the bounds of the variables in two failure regions be used

to measure their "closeness"? How can the similarities of one pair of failure regions be compared relatively to the similarities of another pair when the two pairs are completely dissimilar?

In order to make the problem tractable, it was assumed that the failure regions would have Gaussian distributed behavior. Additionally, it was assumed that all variables affected failure region behavior in the same way. This allowed relationships between failure regions to be identified by the number of variables their bounds had in common.

#### **D. DESCRIPTION OF THE EXPERIMENT**

The empirical data for this study come from a set of programs published in a previous study. Shimeall and Leveson wrote a functional specification for a combat simulation program. Eight pairs of undergraduate students independently wrote programs based on this specification. The eight programs were then extensively tested (Shimeall and Leveson, 1991). The failure regions for the known faults in these programs have been identified using Bolchoz's method.

The problem is analyzed with graph theory techniques. Failure regions are modeled as nodes in a series of graphs. The relationships between the failure regions are modeled as edges. Edge weights are developed based on how many variables two failure regions share as well as the context of the variables within the failure regions. The single-link clustering method is used to study how failure regions tend to form clusters (Jain and Dubes, 1988, p. 70). The clustering tendencies provide insight into which types of failure region-variable behaviors may provide useful information for fault detection.

#### **E. OVERVIEW OF THE THESIS**

Chapter II gives a more extensive literature review of software testing in general and failure region analysis in particular. Chapter III describes how the data were converted into graphs and discusses the details of graph theory and cluster analysis that apply to this study. Chapter IV describes the methods of analysis and the results of the analyses. Finally, Chapter V summarizes the conclusions that can be drawn from the results and offers directions for further research.

## **II. BACKGROUND AND RELATED WORK**

This chapter reviews software testing definitions and methods for dealing with software failures. It then discusses theories of software testing, concentrating on models that are germane to this work. Next it presents previous work in the area of failure regions analysis. Finally, it reviews the basis for the cluster analysis techniques that are used in the experimental portion of this study.

### **A. SOFTWARE TESTING**

#### **1. Faults and Failures**

Software developers realized long ago that virtually all software is faulty. However, faulty programs do not always fail. While this may be fortunate from the standpoint of the user, it is troublesome from the standpoint of the tester. The telephone system example cited in the first chapter demonstrated that a program may run correctly for an indefinite period of time before it fails. It also showed that just because a fault goes unnoticed it does not mean that the failure will be insignificant. A great deal of money and productivity were no doubt forfeited by customers who suffered the loss of their telephone service.

If software developers concede that their software contains faults and if they desire to ensure that those faults do not result in software failures, then the question is how to deal with the faults. There are two possible approaches: either they must find the faults and eliminate them or they must develop methods of tolerating the faults. This thesis deals with the fault-elimination approach.

#### **2. Software Fault Elimination**

The goal of software fault elimination is to find every fault in the software and remove it, thereby producing a fault-free program. There are numerous methods of fault elimination. The literature on these is extensive and will not be reviewed here. Myers (Myers, 1979) and Beizer (Beizer, 1990) both give excellent

surveys of these methods. The discussion here will concentrate on fault-based testing.

**a. *Two Different Approaches to Software Testing***

Myers claims that since software contains faults and since the purpose of software testing is to eliminate faults, then the only successful test is one that finds a software fault (Myers, 1979, pp. 4-7). In other words, if the program runs correctly on a given test, then that test failed. This approach requires a somewhat destructive mentality; the tester is trying to break the program and he is disappointed if he cannot. Many software testers have subscribed to this theory.

The difficulty with Myers' theory is that there is no clear criterion for termination of testing. Neither tests that succeed nor tests that fail under this theory provide any information about either the presence or the absence of other faults in the software.

Morell offers a more constructive theory of testing (Morell, 1990). The difference in his approach is not so much in the tests that are run as in the information that can be gleaned from the tests. Under this theory, a test that fails by Myers' definition may still yield valuable information about which faults specifically cannot exist in the program. The advantage of this theory is that a criterion for completion of testing is available. The tester specifies the faults that he wishes to ensure are not present; he then tests to show that those faults are not present. The danger, of course, is that the tester may fail to specify faults that are, in fact, present in the software.

Methods based on Myers' theory have primarily been concerned with establishing the necessary conditions for a fault to cause a failure. An example of these conditions would be all-statements coverage. However, in order to ensure that a fault causes a failure during testing, both the necessary *and* the sufficient conditions must be met. The necessary conditions only guarantee that a fault will be executed. The sufficient conditions, on the other hand, guarantee that if a fault is executed then it will produce a failure. This is where Morell's theory offers

advancement over previous theories. The next section outlines a theory of test data selection that aims at being both necessary and sufficient.

***b. A Theory of Test Data Selection***

Goodenough and Gerhart first presented the idea of selecting test data that *guarantee* detection of faults. They called a test data set reliable if it uncovered a given fault consistently and valid if it was capable of detecting every error in the program. They called a test set complete if it was both reliable and valid. They suggested using condition tables derived from the program specification for selecting test data. (Goodenough and Gerhart, 1975)

Weyuker and Ostrand pointed out that while Goodenough and Gerhart's theory provided valuable insight on the properties that test data should have, it did not tell the tester how to find such data. In general, it is difficult to devise tests that meet Goodenough and Gerhart's definition of completeness. Weyuker and Ostrand suggested a more pragmatic goal for testing, namely, proving the absence of specified faults rather than all faults. They proposed to do this by using revealing subdomains. A revealing subdomain is a subset of a program's input domain that contains only inputs that are guaranteed to reveal a fault. In other words, revealing subdomains provide the necessary and sufficient conditions for producing failures from specified faults. (Weyuker and Ostrand, 1980)

Weyuker and Ostrand generated revealing subdomains by intersecting two input domain partitions. The first partitioning was into sets that caused a specific path or family of related paths to be executed. They called these path domains. These partitions describe how the program actually treats the input domain. The second partitioning was based on program specifications, algorithms, and data structures. They called this the problem partition. These partitions describe how the program should treat the input domain based on the desired function of the program. The intersection of these two partitions produced sets that were characterized by the conjunction of the path conditions and the problem conditions. These are the sets they used for test data selection. Since ideally the

two partitions should agree, intersections where they do not agree are probably fruitful places to search for failure producing inputs. (Weyuker and Ostrand, 1980)

Richardson and Clarke proposed a method similar to Weyuker and Ostrand's. They partitioned the input space into subdomains using information from both the program's specification and its implementation. They then proposed using symbolic execution to determine if the implementation agreed with the specification. (Richardson and Clarke, 1981)

Richardson and Thompson developed the RELAY model of fault detection based on an earlier version of Morell's fault-based testing theory. A potential failure is originated when a fault is executed. This is the necessary condition for failure. The potential failure is then relayed through the program by computational and data flow transfers until it is manifested as an output error [failure]. The computational and data flow transfers are the sufficient conditions for failure. The failure must be both originated and relayed or it will not be revealed. Thus, this model provides a practical framework for selecting test data that are both necessary and sufficient for guaranteeing fault detection. (Richardson and Thompson, 1988)

### ***c. Mutation Testing***

The works of Morell and of Richardson and Thompson are adapted from mutation testing (DeMillo, et al., 1978). The idea of mutation testing is predicated on two assumptions. The first is the competent programmer assumption; it is assumed that the software is only "slightly" incorrect. For example, it is assumed that a numerical integration algorithm is not used in place of a differentiation algorithm. Although the assumption seems reasonable, it cannot be verified or for that matter even quantified. The second assumption of mutation testing is the coupling effect; that is, that tests that detect simple faults will also be sensitive to more complex faults. This effect is further discussed in the cluster analysis section below.

The basic method of mutation testing is to try to identify the classes of faults that might exist in the software. Perhaps the designer indexed an array with

the wrong loop counter or the programmer substituted a Boolean OR for a Boolean AND. Mutations of the program are generated from the identified classes of faults. Test data are then sought that will distinguish the mutations from the original program. Mutations that survive the testing are either functionally equivalent to the original program or the test data are not sensitive enough to make the distinction.

#### ***d. Partition Testing***

All the testing theories and methods that have been discussed here fall under the general category of partition testing. The primary characteristic of partition testing is that the program's input domain is divided into subdomains. The tester builds his test set by selecting elements from each subdomain. Partition testing ranges from random testing to exhaustive testing. In the former, there is one partition, namely, the entire input space. In the latter, there are as many partitions as there are elements in the domain. Mutation testing is partition testing in that it divides the domain into partitions that distinguish the various mutants.

Weyuker and Jeng examined partition testing strategies analytically. They showed that, in general, arbitrary partitioning strategies may provide results that are either better or worse than random strategies. (They used partitioned to mean more than one subdomain.) They also showed that if an appropriate method exists for refining partitions, then improvement of the performance of partitioning strategies over random strategies can be guaranteed. While Weyuker and Jeng present no specific strategy, their results suggest that refinement should be fault-based, i.e., that partitions should be designed with particular faults in mind. (Weyuker and Jeng, 1991)

In summary, most testing strategies guess at the nature of the faults that might be present and then try to develop test sets to uncover the hypothesized faults. This might be characterized as an outside-to-inside approach. Little study has been done to determine how faults really behave. This study has the goal of determining actual fault characteristics that may be useful in locating faults. This might be termed more of an inside-to-outside approach to testing.



### **3. Failure Regions**

A subset of the program domain is associated with every fault in a program. Sets of bounds delimit this subset, one set of bounds corresponding to each of the variables in the domain. These bounds identify the values that the program variables must assume in order for that specific fault to cause a program failure. Every variable must be within its specified bounds before that fault will produce a failure. Ammann and Knight called the subset of the domain associated with a fault its failure region (Ammann and Knight, 1988).

Ammann and Knight used failure regions to develop an approach to software fault tolerance called data diversity. They suggested that for many program variables there is a set of values that will produce equivalent program behavior. If a fault produces a failure and if there is an equivalent value for the offending variable that lies outside the failure region, then failure can be avoided by substituting the equivalent value. Data diversity is a fault-tolerance technique rather than a fault-elimination technique. (Ammann and Knight, 1988)

Bolchoz developed an analytical method for identifying failure regions (Bolchoz, 1990). He described three conditions that are required for a fault to produce a failure. First, all the conditions for the fault to be reached must be met, e.g. appropriate procedure calls and program branches. Second, the fault must be executed in a way that produces an error or an erroneous intermediate result. Finally, the error must be propagated to a final result without being masked by subsequent processing. The failure region of a fault is the set of data values that satisfy the conjunction of these three conditions. The difference between this method and that of Weyuker and Ostrand is that this method identifies conditions for execution of a specific fault that is already known to exist while their method identifies conditions for where hypothesized faults are likely to exist. Shimeall, et al., showed that, under certain assumptions, Bolchoz's method provides the necessary and sufficient conditions for a known fault to produce a failure (Shimeall, et al., 1991).

Voas and Morell explored an idea similar to failure region analysis. They called it propagation and infection analysis. They studied the sensitivity of programs to faults by executing the programs rather than by examining the program specification and implementation. They called the probability that a fault will be executed on a randomly selected input the execution rate. The probability that the fault will infect subsequent data states after the error occurs is the infection rate and the probability that the fault will persist to manifest a program failure is the propagation rate. They suggested empirical methods for estimating these rates. They used the conjunction of these individual rates to predict the program's failure rate. (Voas and Morell, 1989)

Failure regions have been used to provide insight into the necessary and sufficient conditions for revealing specific faults and for understanding how specific faults behave in isolation. The study presented in this thesis is the first to collect information on how faults or failure regions are related to each other. Failure regions offer a mechanism for identifying common features among faults. Faults that have similarities in their failure regions might be expected to exhibit similar behaviors when they cause a failure. This thesis explores the similarities and differences between the failure regions of known faults in the same program with the goal of better understanding fault behavior.

## **B. CLUSTER ANALYSIS**

### **1. Definitions**

Much scientific study is based on the classification of objects according to perceived similarities. Cluster analysis is the study of how to build a formal basis for this activity of classification that humans perform almost instinctively. Although the idea of deciding when objects are similar to each other may seem intuitively obvious, researchers have had difficulty in agreeing on a formal definition of a cluster. One definition that fairly well describes the analysis performed in this thesis is: "Clusters may be described as connected regions of a multi-dimensional space containing a relatively *high density* of points, separated from other such regions by a region containing a relatively low density of points." (Jain and Dubes, 1988, p. 1)

## **2. The Basis for Using a Cluster Analysis Approach**

Myers cites anecdotal evidence that the probability of the existence of undiscovered faults in a given section of code is proportional to the number of faults already found in that section (Myers, 1979, p. 16). He calls this tendency error [fault] clustering. Myers is speaking specifically of the proximities of faults to each other in the code, e.g., two sequential statements.

The coupling effect is an idea that is similar to fault clustering. Offutt conducted an empirical study of the validity of the coupling effect. He tested programs that contained automatically generated first-order mutants. He then used the same data sets to test programs that contained second-order mutants that were generated from the first-order mutants. His results offer convincing evidence that any test that is sensitive to "simple" faults will also detect more "complex" faults. (Offutt, 1989)

Mutation testing uses the assumption that there are relationships between faults as a basis for the technique. However, the approach tries to find faults by random (or exhaustive) generation of mutants; this is a rather computationally intensive approach. This thesis explores the idea of identifying the specific relationships that cause fault clusters. Specifically, common features of failure regions from the same program are identified. Failure regions are directly linked to specific faults. Thus, knowledge about these common features may raise the probability of predicting the locations of undiscovered faults based on their relationships to faults that have already been found.

## **3. Cluster Analysis Techniques**

### ***a. A Graph Theory Approach***

The discussion in this section derives from Godehardt's presentation of graphs as structural models and their use in cluster detection (Godehardt, 1988). The discussion is specific to failure regions modeled as nodes and relationships between the failure regions modeled as edges. It is assumed that the reader is familiar with the concepts of graph theory. Definitions of graph theory terms are presented for reference in Appendix A.

The connectivity (edge-connectivity) of a graph gives a qualitative idea of both the nature and the strength of relationships between failure regions. If bridges or cutnodes exist, it may be possible to find blocks in a graph whose connectivity (edge-connectivity) is large relative to that of the graph itself. Even if there are no bridges or cutnodes, a graph that is  $n$ -connected ( $n$ -edge connected) for small  $n$ , e.g. 2 to 4, may still contain significant subgraphs that have relatively larger connectivity (edge-connectivity.) Such blocks or subgraphs would suggest that there are groups of failure regions that are strongly related to each other but only weakly related to other failure regions. If the graph is disconnected then both the absence of relationships between failure regions in different components and the presence of relationships between failure regions within components is emphasized.

The diameter, radius, and center of a subgraph indicate how intricately the failure regions are related. If the diameter is one or two then every pair of failure regions is either directly related or both regions in the pair are related to the same failure region. If the diameter is large but the radius is small, then the center of the graph is a subgraph that has relationships analogous to the supergraph with a small diameter.

Relationships may also be modeled with a multigraph. Each graph in the multigraph has the same node set, but the edge sets are based on different criteria. In general, each graph in the multigraph has blocks containing different sets of nodes. If failure regions appear in two or more blocks across the multigraph, this might suggest how two different clustering criteria were related to each other. These failure regions might also be important in characterizing variables that lead to certain faults.

#### ***b. A Traditional Clustering Approach***

The goal of cluster analysis is to identify groups of objects that have similar characteristics. Most traditional clustering algorithms (as opposed to the graph theory methods described above) work on some variation of the following method:

1. For the object that is to be placed in a cluster, find the single object that is "closest" to the object of interest, and put those two objects in the same cluster.
2. If there is no "close" object, then start a new cluster.

In other words, the clustering is essentially based on an object's relationship to its closest neighbor.

There are two basic classifications of clustering techniques that follow this algorithm: partitioned and hierarchical. Partitioned clusters require every object to be in exactly one cluster. The researcher must decide a priori at what distance an object is too far away from its neighbors to be included in their cluster. This approach assumes that objects in different clusters are completely dissimilar.

The hierarchical approach assumes that if the restrictions for comparison are relaxed sufficiently (e.g., to no restriction at all), then no two objects are absolutely dissimilar. This method starts by forming clusters with strict criteria and then allows the clusters to merge as the criteria are relaxed. When the clustering criteria have been relaxed sufficiently, all the objects will form one cluster.

One difficulty in applying these methods to the current problem is in determining when two failure regions are close to each other. The sample space is heterogeneous and the relationships between the failure regions are ordinal. Both of these factors make the idea of Euclidean distance meaningless. Some other measure of "distance" between failure regions is required. The approach used in this study is described in detail in the next chapter.

The clustering method used in this study is a hierarchical method called single-link clustering. The method uses a threshold graph to construct the clusters. This method is also described more fully in the next chapter.

### **C. CONCLUSION**

This chapter has described the background needed to support this study of failure region analysis. A reliable method for finding faults in software needs to be developed. An important element of a reliable testing method is its ability to estab-

lish both the necessary and the sufficient conditions for a fault to be revealed. Failure regions developed using Bolchoz's method have been shown to establish these conditions for known faults. If relationships between failure regions can be characterized, then the failure regions of detected faults may provide information about where to find still more faults. One step towards characterizing these relationships is to determine the clustering tendencies of failure regions. The next chapter describes the experiment used for studying failure region clusters.

### **III. EXPERIMENTAL DESIGN**

#### **A. INTRODUCTION**

This study analyzed similarities and differences between failure regions. The primary goal of this research was to develop a technique for empirically examining failure regions to determine what relationships exist. A secondary goal was to characterize the relationships. A set of programs written to the same specification were taken from a previous study (Shimeall and Leveson, 1991). The faults in these programs provided the failure regions used in this study. Patterns of variable usage were identified in these failure regions. Graphs based on this analysis used nodes to represent failure regions and edges to represent relationships based on the context and frequency of variable usage. Clustering patterns and tendencies among the failure regions were identified from these graphs.

This chapter describes the data that were used for the study and how the data were reduced to a form useful for analysis. The methods of generating the graphs, including the edge weights, are presented. Finally, cluster analysis techniques are discussed.

#### **B. DESCRIPTION OF THE DATA**

Shimeall and his students are using a set of eight programs in an ongoing series of software testing studies. Shimeall wrote a functional specification for a combat simulation program. Eight pairs of undergraduate students separately wrote programs based on this specification. Shimeall then extensively tested the programs using code reading, assertions, testing, and voting. The numbers of known faults in each of the various programs range from as few as 25 to as many as 50. (Shimeall and Leveson, 1991).

Bolchoz developed a method for determining the failure region of a fault based on the conditions that must be met for that fault to cause a failure (Bolchoz, 1990). Shimeall used Bolchoz's method to generate the failure regions of the faults in four

of the eight programs. Appendix B contains the failure region definitions for Version 1 of the program as an example. The complete set of failure region definitions is contained in a separate report (Shimeall, 1991). Table 3.1 gives a profile of the faults and failure regions by program version. The term dimensions is used to refer to either input variables or to predicates composed of input variables. The numbers of input variables exceed the numbers of dimensions because there are several input variables that appear only in the variable predicates. These predicates are discussed further in the next section.

**TABLE 3.1: PROFILE OF FAILURE REGIONS**

	Versions			
	1	2	3	4
Known faults	26	30	46	36
Noncoincident regions	23	26	38	27
Total dimensions	53	48	52	53
Mean dimensions per region	7.52	5.68	6.22	7.86
Std. dev. dimensions per region	5.72	2.98	3.93	3.20
Total input variables	69	72	75	67
Mean input variables per region	38.83	38.56	42.76	52.33
Std. dev. input variables per region	27.85	27.77	25.23	4.80

### **C. DATA REDUCTION**

The first step in developing a strategy for exploiting relationships between failure regions was to determine how to identify the relationship. Failure regions are defined by bounds on the various program variables. This suggested that the relationships sought in this study might also be described in terms of these variable



bounds. Variables may be considered according to their syntax or their semantics. Syntax deals with whether the variable is used legally within the constraints of the language and the program. Semantics deals the meaning of the variable in a specific context. This study considered both syntax and semantics.

### **1. The Use Of Predicates**

As Shimeall derived the failure regions, he noted that some variables were used under commonly occurring conditions. The conditions were frequently related to semantic contexts in the program specification. When these conditions were noted, predicates were substituted for individual variables in order to identify the semantic context of the failure region.

Predicates were treated in the same way as individual variables during the analysis. There were two reasons for choosing this approach. The first was that even though many of the same variables participate in the various predicates, the predicates are semantically different. Preserving the semantic contexts of these sets of variables within their respective failure regions helps to clarify the relationships between the failure regions.

The second reason for using the predicates rather than their component variables was that most of the predicates involved numerous variables. Edges in the graphs were determined by how many variables two failure regions' bounds had in common. Since at least one predicate occurred in most of the failure regions, using only the individual variables could have resulted in complete, or nearly complete, graphs. This might have obscured interesting results.

The problem with leaving the predicates intact was that the predicates are essentially semantic. On the other hand, individual variable incidence is primarily syntactic. This mixing of semantic and syntactic forms in the same analysis could lead to some distortion, especially since the decision of when to condense a set of bounds into a predicate was somewhat arbitrary.

Thus, while it was recognized that some distortion would probably result from either treatment of the predicates, it seemed that treating them in the same manner as individual variables was more likely to filter some of the noise out of the

graphs and draw more attention to the useful differences and similarities of the failure regions. Hereinafter, variables and predicates will be referred to collectively as failure region dimensions.

## **2. Analysis of Dimension Participation in Failure Region Bounds**

Each pair of failure regions within a given version was compared. For each pair, each dimension was classified as participating in one of the following ways:

1. The dimension appeared in both regions' bounds in exactly the same way. For example, in failure regions 1.3 and 1.4, `Params.NumWEvents` participates in the bounds as an index to the same dimension (see Appendix B). This type of participation was termed *Identical*.
2. The dimension appeared in both regions' bounds but was not *Identical*. This type of participation was termed *Coincidental*.
3. The dimension did not participate in the bounds of either of the regions in the pair. This type of participation was termed *Nonbounding*. What this type of participation really means is that the bounds that this dimension place on the failure region are no more restrictive than the entire range of values that this dimension can assume.

The *Identical* and *Coincidental* dimensions are referred to collectively as the *Composite* dimension.

(Initially, an attempt was made to identify dimensions that had similar behavior between two failure regions. For example, if the same dimension participated in an inequality in both failure regions but the inequalities were not *Identical*, this might have been considered *Similar*. However, subsequent analysis showed that the *Similar* dimension offered no useful insight and *Similar* was discarded as a separate dimension classification.)

Dimensions that were *Nonbounding* for all failure regions in a given version were discarded from that version's matrix. This significantly reduced the size of the matrices since there were 127 variables, besides the predicates, that could potentially participate in the bounds. The various versions studied here actually use from 48 to 53 dimensions in the bounds of their failure regions.

### **3. The Failure Region-Dimension Incidence Matrices**

The results of the dimension evaluations were placed into incidence matrices with dimensions on the rows and failure regions on the columns. These matrices are in Appendix C.

In the matrix for Version 1, the entry in column 1.10 for `Army[].Squadrons` is I10. This entry indicates that the participation of `Army[].Squadrons` in failure region 1.10 is Identical to itself and is not Identical to its participation in any of the first nine failure regions. Coincidental behavior between two failure regions is indicated if they both have an entry, but they are not Identical to the same failure region. A blank entry in the matrix indicates that the given dimension is Nonbounding for the given failure region.

Each entry in the matrix is referenced to the lowest numbered failure region to which that dimension is Identical. As an example, both columns 1.10 and 1.11 contain the entry I1 for the dimension `NArmy[]`. This means that the participation of `NArmy[]` in both of these regions is Identical to that in region 1.1. This is clearly a transitive property, so the participation of `NArmy[]` in region 1.10 may be inferred to be Identical to its participation in region 1.11.

Both the failure regions' definitions and the failure region-dimension incidence matrices were generated manually. Because of this, some errors have undoubtedly been made. However, the numbers of distinct failure regions in the various versions used in this study range from 21 to 37. Thus the smallest graph could have as many as 210 edges while the largest could have as many as 666. If the errors are few, the affect on the validity of the qualitative results should not be significant.

### **4. Failure Region Graphs**

Graphs were generated from the failure region-dimension incidence matrices for each version of the program. Each failure region was treated as a node. Weighted edges between the nodes were based on the numbers of dimensions the failure regions had in common as well as how those dimensions participated in their bounds.

The edge weights were calculated using the program listed in Appendix D. This program takes the failure region-dimension incidence matrix as an input. It identifies the value associated with each failure region-dimension pair, i.e. 1 or blank. It also identifies the associated failure region number, i.e. the number following the 1. The program stores these values in an array indexed by dimension and failure region numbers.

The program uses the failure region-dimension array to count the number of occurrences of Identical and Coincidental dimensions for each pair of failure regions. It also counts the total number of dimensions that appear in the bounds of at least one of the failure regions in that pair. The program calculates the edge weighting coefficients from these counts. (These coefficients are described in subsection 5 below.) Finally, the program lists:

1. the edges, in descending order of their coefficients,
2. the coefficient and the dimension counts associated with each edge (i.e. and coefficient numerator and denominator), and
3. the nodes, in order based on their largest incident edge.

These graphs are presented in tabular form in Appendix E.

## **5. Determination Of Edge Weights**

### ***a. Separate Analysis of Identical and Coincidental Data***

The data for this study are essentially ordinal, namely, in descending order: Identical, Coincidental, Nonbounding. Relative values cannot be assigned to data that are inherently ordinal; thus, there is no way to develop a single edge weight that accurately represents the relationship between two failure regions. Because of this, three separate graphs were developed for each version of the program.

The first graph considered only Identical bounds. The second graph considered only Coincidental bounds. The third graph lumped the Identical and Coincidental dimensions together to form the Composite dimension. This third graph was developed to test whether splitting the dimension behaviors into Identical and Coincidental had produced any artificial affects. This, then, resulted

in three sets of binary data: Identical or not, Coincidental or not, and Composite or not.

***b. Selection of the Weighting Coefficient***

Two different coefficients were considered for determining the values of the edge weights. Both coefficients give an indication of how closely related two failure regions are. The first was the simple matching coefficient, given in Equation 3.1 (Jain and Dubes, 1988, p. 17). The numerator of this coefficient is the sum of the number of dimensions that are Composite for both regions and the number of dimensions that are Nonbounding for both regions. The denominator of the simple matching coefficient is the total number of dimensions.

$$S(m, n) = \frac{a_{00} + a_{11}}{a_{00} + a_{01} + a_{10} + a_{11}} \quad (\text{Eq 3.1})$$

where:

$S(m,n)$  - simple matching coefficient for regions  $m$  and  $n$ .

$a_{00}$  - number of dimensions that are Nonbounding for  $m$  and  $n$ .

$a_{01}$  - number of dimensions that are Composite for  $m$  but not  $n$ .

$a_{10}$  - number of dimensions that are Composite for  $n$  but not  $m$ .

$a_{11}$  - number of dimensions that are Composite for  $m$  and  $n$ .

The simple matching coefficient assigns as much importance to Nonbounding dimensions as it does to Composite dimensions. The nonparticipation of a given dimension in a failure region simply means that the fault can result in a failure regardless of the value of that dimension. The primary goal of this study was to determine if failure regions can be used to identify dimensions of interest for software testing. Therefore, it is not particularly useful to know that the value of a dimension is irrelevant when the fault causes a failure. For the purposes of this study, the participation of a dimension in the bounds of a failure region is more significant than the nonparticipation of a dimension.

The second coefficient considered was the Jaccard coefficient,  $J(m,n)$ , given in Equation 3.2 (Jain and Dubes, 1988, p. 17). The  $a_{ik}$  for this coefficient have

the same meaning as those for the Simple matching coefficient. The numerator of this coefficient is the number of dimensions that are Composite for both failure regions. The denominator is the number of dimensions that are Composite for at least one of the regions. This coefficient places a heavier emphasis on dimension participation than on nonparticipation. The Jaccard coefficient that was used in this study.

$$J(m, n) = \frac{a_{11}}{a_{11} + a_{01} + a_{10}} \quad (\text{Eq 3.2})$$

The Jaccard coefficient had to be modified to analyze the Identical and Coincidental data individually. The reason is that for the numerator, the condition to be satisfied is not just Composite but specifically Identical or Coincidental. In other words, for the graph of Identical values, the numerator of the coefficient is only the number of dimensions that are Identical between the two regions, as is shown in Equation 3.3. The Coincidental data are treated similarly in Equation 3.4.

$$I(m, n) = \frac{i_{11}}{a_{11} + a_{01} + a_{10}} \quad (\text{Eq 3.3})$$

where:

$I(m,n)$  - modified Jaccard coefficient for Identical dimensions

$i_{11}$  - number of dimensions that are Identical in regions m and n

$$C(m, n) = \frac{c_{11}}{a_{11} + a_{01} + a_{10}} \quad (\text{Eq 3.4})$$

where:

$C(m,n)$  - modified Jaccard coefficient for Coincidental dimensions

$c_{11}$  - number of dimensions that are Coincidental in regions m and n

## **D. CLUSTER ANALYSIS**

### **1. Clustering Method**

Two methods were considered for use in identifying failure region clusters. The first method was to look for  $k$ -connected subgraphs (Godehardt, 1988). This method requires searching for all possible paths between every pair of nodes in the graph. This is an NP complete problem. Additionally, in a weighted graph, the problem must be solved for each threshold value of interest.

$K$ -connected subgraphs were not used for two reasons. The first was that the method is too detailed for exploratory analysis. It is more suited to identifying specific clusters in data where the clustering behavior is already well understood, i.e., where the range of  $k$  is fairly well estimable. The second reason for not using this method was its computational complexity. Again, this inhibits exploratory analysis.

The second method was adapted from Jain and Dubes (Jain and Dubes, 1988, p. 70). This method, called Single-Link Clustering, is also based in graph theory but follows more closely the traditional ideas of cluster analysis. Clusters are developed by adding edges to the graph in the order of their relative weights. As the weight threshold becomes less restrictive, more edges are added to the graph.

The addition of a new edge to a graph can have one of two results. The first is that the edge may connect two nodes that were already connected by edges at more restrictive weight thresholds. Edges such as these have the effect of strengthening existing clusters. The other result a new edge may have is to merge two components in the graph. If one of these components has multiple nodes, that edge has increased the size of a cluster. If both components are singleton nodes, the edge has initiated a new cluster. (While a singleton node is technically a one element cluster, the discussion here uses cluster to mean a grouping of two or more nodes.)

The modification to the Single-Link method as described in Jain and Dubes was that the requirement that no two edges have the same weight was relaxed. This modification was reasonable since the goal of this study was not to

identify specific failure regions in specific clusters; nor was it the goal to identify the specific order in which failure regions were added to clusters. Rather, the goal was to identify whether there was even a tendency for failure regions to cluster in a way that was useful for developing software testing strategies.

One note should be made regarding the use of the Jaccard coefficient in conjunction with this cluster analysis method. Most clustering methods assume that a smaller edge weight indicates nodes that are more similar to each other, i.e. more strongly clustered. This idea comes from the fact that edge weights are frequently derived from Euclidean proximities. For edge weights based on the Jaccard coefficient, however, the closer the coefficient is to one, the more alike the failure regions are. (The range of the coefficient is from zero to unity.) This does not invalidate the clustering method; it merely means that edges are added to the graph by lowering the threshold rather than by raising it.

## **2. Hierarchical Vs. Partitioned Clusters**

The clustering method used in this study produces an hierarchical clustering rather than a partitioned one. This is the type of clustering that was desired since it was not clear that failure regions should necessarily belong to exactly one cluster. Indeed, since the goal of this study was to determine if knowledge about one failure region can be used to find other failure regions, hierarchical clustering is more desirable than partitioned clustering.

If it is the case that failure regions have a strong hierarchical clustering tendency, then at least two different ways of exploiting the clusters are suggested. First, the stronger (i.e. more restrictive threshold) clusters may provide a method to find the other failure regions within those clusters. Second, the potential exists to "bootstrap" from one strong cluster to another under the right conditions. This would involve identifying the types of dimension participations that result in the edges that appear at the less restrictive threshold values.

## **E. CONCLUSION**

This chapter has detailed the procedures followed in analyzing the data used for this study. The known faults in four versions of the same program were used to



develop the failure regions for those faults. The failure regions were analyzed for Identical and Coincidental dimension behavior. The frequency of these types of behavior was then used to develop weighted graphs. These weighted graphs provide a means for evaluating the tendency of failure regions to form clusters. The analysis of these clusters is the subject of the next chapter.

## **IV. EMPIRICAL RESULTS**

### **A. INTRODUCTION**

This chapter presents the results of the experiment discussed in the previous chapter. Before proceeding, however, some caveats should be noted. First, student programmers produced the software used for this study. While these students may have had significant experience in programming, they cannot, in general, be classed with professional programmers. Fault populations produced by professional programmers may vary significantly from those of student programmers. Additionally, the programs were all for the same application, namely, a battle simulation. Different types of applications may also produce significantly different distributions of faults.

There are also some limitations that result from the experimental design. Only one method of quantifying the relationship between two failure regions was studied, namely, a modified Jaccard coefficient. Additionally, only threshold graphs were used for cluster analysis. The narrow focus of the design may impose an artificial structure on the data. Using only one analysis method may also obscure important features of the data or highlight insignificant features.

With these limitations in mind, and realizing that extensibility of the results beyond this one application has yet to be established, the results still provide useful insight into how faults are related to each other. The next section describes how the data are presented. After that, notable characteristics of the data and the validity of these characteristics are discussed. Finally, the results are interpreted with a view towards software testing applications.

### **B. DATA PRESENTATION**

Dendograms are the typical method of presenting data for hierarchical cluster analysis. However, the goal of the cluster analysis in this study was not to identify specific failure region clusters in specific programs. Rather, the goal was to deter-

mine whether failure regions even have a tendency to form clusters. For this reason, histograms were used instead of dendograms. The advantage of histograms is that they are easier to use in comparing the behavior of several populations. Dendograms are more useful for analyzing a single population.

For each program version, two histograms were constructed for each dimension type. The first histogram shows how many edges are added to the graph in each interval of the Jaccard coefficient. In the second histogram, the column in each Jaccard coefficient interval shows how many nodes have their largest incident edge in that interval. These histograms are presented in Appendix F.

The first histogram presents additional information. The total column height in each interval shows the number of edges that have weights in that interval. The column is divided into two parts. The black part, labeled "Between Newly Connected Nodes," shows the numbers of edges that are incident on nodes that had no incident edge in a higher threshold interval. This information corresponds directly to the numbers of nodes shown in the second histogram. The gray part, labeled "Between Previously Connected Nodes," shows the numbers of edges that are incident on nodes that did have an incident edge in a higher threshold interval.

The edges were divided into "Between Newly Connected Nodes" and "Between Previously Connected Nodes" to help clarify the types of clustering behavior that the failure regions were exhibiting. The former category helps determine the numbers of edges involved in merging pairs of singleton nodes into new clusters or adding singleton nodes to a cluster. The latter category helps determine when previously defined clusters are being strengthened or are merging. While "Between Previously Connected Nodes" does not distinguish between edges added within a cluster and edges added between clusters, this is not important because it would not provide additional information about whether failure regions tend to form clusters, which is the primary goal of the cluster analysis. Although the strength and size of clusters would be important in practical software testing applications of failure region clusters, the more important questions for this study are: how many nodes are in *some* cluster and are the nodes added to the cluster at a statistically significant threshold level?

The abscissae of the histograms are labeled with the Jaccard coefficient decreasing from left to right. The reason for this convention is that cluster data are typically presented so that the more significant edges are to the left in the histogram or dendogram. This requires the largest value of the Jaccard coefficient to be presented at the left.

The histograms are divided into intervals of 0.05. In general, the data included in an interval are strictly less than the upper limit of the interval and greater than or equal to the lower limit. There are two exceptions: data in the uppermost interval are less than or equal to unity; data in the lowermost interval are strictly greater than zero. The reason for the first exception is obvious. The reason for the second exception is that edges of zero weight represent the absence of a relationship between two failure regions while nonzero edges represent the presence of some relationship, however weak. Inclusion of the zero weight edges might have skewed the histograms and lead to false conclusions about failure region clustering tendencies.

In several cases, two or more distinct faults shared identical failure regions. When this occurred, the failure region was considered only once in constructing the histograms and the graphs. The reason is that if faults share identical failure regions, any test that reveals one of the faults will reveal all of them. The goal of this study is to find a method to reveal new failure regions rather than redundant ones.

## **C. DATA ANALYSIS**

### **1. Notable Characteristics**

Analysis of the histograms suggests that there is indeed a tendency for failure regions to form clusters. For the identical dimensions, all four versions' histograms exhibit small groups at relatively large thresholds. These groups correspond to several small and unconnected clusters being formed. Over half of the nodes in the graphs have at least one incident edge in these higher threshold intervals. This is as opposed to many edges being added between just a few nodes.

The behaviors of the Coincidental and Composite dimensions are broadly similar to that of the Identical dimensions. However, there appears to be a difference in how the clusters grow. (This is discussed further in Section D.) Additionally, there seems to be more variation in the behavior between the versions for Coincidental dimensions as opposed to Identical dimensions. It is difficult to judge whether there are, in fact, significant differences here since there are only four versions to compare.

## **2. Data Validation**

In order to verify that the noted characteristics could not be attributed to random behavior or to the experimental method, the experimental data were compared with a random population of regions. The null hypothesis to be tested by this comparison was: there is no difference in behavior between the experimental population of failure regions and a population of regions bounded by arbitrarily selected conditions occurring in the source code. Rejection of the null hypothesis indicates that clustering a behavior of the faults rather than the application studied or the analysis technique employed.

Failure regions are bounded by conditions that either arise directly from the program source code or are synthesized from the source code. The random regions were bounded by conditions that were randomly extracted from the Gold version of the program in Shimeall and Leveson's study (Shimeall and Leveson, 1991). The Gold version was used to ensure that the random regions were not biased in favor of one of the test versions. The conditions were selected from a text file using the UNIX library function `random`. The distribution of the numbers of conditions in the random regions was selected to reflect the number of dimensions in the experimental failure regions.

Two populations of random regions were used in order to match the sizes of the experimental populations. A 20 region set was used to approximate Versions 1, 2, and 4; a 40 region set was used to approximate Version 3. The 20 region set was a subset of the 40 region set. These sets are referred to as R20 and R40, respectively. A statistical profile of the random regions is given in Table 4.1. The

mean number of input variables in the random regions is smaller than for the experimental regions since the random regions contain no predicates. If the predicates in the experimental regions are not expanded, the mean number of dimensions of the experimental regions is similar to the mean number of variables in the random regions.

**TABLE 4.1: PROFILE OF RANDOM FAILURE REGIONS**

	R20	R40
Minimum/Maximum variables in a region	3/17	3/17
Total input variables	50	60
Mean input variables per region	8.85	9.98
Std. dev. input variables per region	4.31	4.06
Mean input conditions per region	6.50	6.92
Std. dev. input conditions per region	2.26	2.39

The random regions were treated with the same experimental procedure as the experimental regions. One way analysis of variance (ANOVA) was applied to the four experimental versions and the two random versions for both the edge and node distributions and for each type of dimension. The actual edge weights (as opposed to the histogram distributions) were used in this analysis. Computations were performed with the UNIX|STAT data analysis program oneway (Perlman, 1986). The results of the analysis are presented in Tables 4.2 through 4.7. The column headings are self explanatory except for the last two; P(R20) and P(R40) are the probabilities that the given experimental distribution is the same as the random distribution. These probabilities are based on a Student t test.

The results of ANOVA indicate that the null hypothesis can be rejected. The experimental edge distributions differ from the random edge distributions at better

**TABLE 4.2: IDENTICAL DIMENSION EDGE STATISTICS**

Version	N	Mean	SD	Min	Max	P(R20)	P(R40)
1	57	0.166	0.147	0.031	0.700	<0.005	<0.001
2	42	0.273	0.190	0.062	0.636	<0.001	<0.001
3	236	0.175	0.142	0.040	0.909	<0.001	<0.001
4	58	0.126	0.109	0.048	0.700	<0.1	<0.01
R20	65	0.097	0.055	0.040	0.250	-----	-----
R40	288	0.097	0.064	0.034	0.476	-----	-----
Total	746	0.139	0.123	0.031	0.909	-----	-----

**TABLE 4.3: COINCIDENTAL DIMENSION EDGE STATISTICS**

Version	N	Mean	SD	Min	Max	P(R20)	P(R40)
1	160	0.211	0.122	0.034	0.533	<0.001	<0.001
2	211	0.187	0.105	0.059	0.500	<0.001	<0.001
3	529	0.204	0.128	0.043	1.000	<0.001	<0.001
4	196	0.203	0.142	0.045	0.889	<0.001	<0.001
R20	140	0.128	0.080	0.042	0.389	-----	-----
R40	681	0.154	0.098	0.034	0.500	-----	-----
Total	1917	0.180	0.117	0.034	1.000	-----	-----

**TABLE 4.4: COMPOSITE DIMENSION EDGE STATISTICS**

Version	N	Mean	SD	Min	Max	P(R20)	P(R40)
1	168	0.258	0.179	0.036	1.000	<0.001	<0.001
2	223	0.228	0.179	0.059	1.000	<0.001	<0.001
3	603	0.248	0.177	0.043	1.000	<0.001	<0.001
4	210	0.224	0.173	0.045	1.000	<0.001	<0.001
R20	154	0.158	0.099	0.042	0.500	-----	-----
R40	719	0.185	0.118	0.034	0.667	-----	-----
Total	2077	0.216	0.157	0.034	1.000	-----	-----

**TABLE 4.5: IDENTICAL DIMENSION NODE STATISTICS**

Version	N	Mean	SD	Min	Max	P(R20)	P(R40)
1	23	0.312	0.193	0.000	0.700	<0.005	<0.1
2	26	0.290	0.224	0.000	0.636	<0.05	>0.1
3	38	0.378	0.215	0.000	0.909	<0.001	<0.001
4	21	0.257	0.200	0.000	0.700	<0.1	>>0.1
R20	20	0.168	0.059	0.091	0.250	-----	-----
R40	40	0.243	0.084	0.125	0.476	-----	-----
Total	168	0.283	0.184	0.000	0.909	-----	-----

**TABLE 4.6: COINCIDENTAL DIMENSION NODE STATISTICS**

Version	N	Mean	SD	Min	Max	P(R20)	P(R40)
1	23	0.410	0.129	0.000	0.533	<0.001	<0.1
2	25	0.359	0.128	0.000	0.500	<0.05	>>0.1
3	37	0.484	0.208	0.154	1.000	<0.001	<0.005
4	21	0.503	0.225	0.188	0.889	<0.001	<0.005
R20	20	0.276	0.082	0.125	0.389	-----	-----
R40	40	0.357	0.092	0.182	0.500	-----	-----
Total	166	0.402	0.168	0.000	1.000	-----	-----

**TABLE 4.7: COMPOSITE DIMENSION NODE STATISTICS**

Version	N	Mean	SD	Min	Max	P(R20)	P(R40)
1	23	0.593	0.222	0.000	1.000	<0.001	<0.005
2	25	0.518	0.253	0.000	1.000	<0.01	>0.1
3	37	0.704	0.184	0.167	1.000	<0.001	<0.001
4	21	0.599	0.236	0.231	1.000	<0.001	<0.005
R20	20	0.338	0.112	0.167	0.500	-----	-----
R40	40	0.447	0.120	0.250	0.667	-----	-----
Total	166	0.541	0.221	0.000	1.000	-----	-----



than a 99 percent confidence level with the exception of Version 4's Identical dimensions, which have a better than 90 percent confidence level. Given the exploratory nature of this study and the lack of prior information about the experimental population, a 90 percent confidence level is generally considered acceptable. Thus, if  $P(R) > 0.1$  the experimental data were not considered significantly different from the random regions.

The node distributions had a wider range of variation, but most of the experimental distributions differed from the random distributions with greater than 90 percent confidence. There were some notable exceptions. For the Identical dimension, Version 2 and Version 4 were not significantly different from R40. For the Coincidental dimension, Version 2 did not vary significantly from R40. Finally, for the Composite dimension, Version 2 did not vary significantly from R40.

While Version 3 is clearly different from the random distributions in all cases, there seems to be a contrast between the node distributions of Versions 1 and 2 and Version 4. Version 4 was significantly weaker than either Version 1 or 2 for the Identical dimension while it was significantly stronger for the Coincidental dimension. There is insufficient data to determine whether more random behavior in one dimension leads to less random behavior in another dimension.

### **3. Cluster Formation**

The general shapes of the experimental data histograms are slightly but significantly different from the random data histograms. Specifically, the small groups of edges and nodes at higher coefficient thresholds are absent in the random distributions. However, the experimental distributions appear to be overtaken by random behavior below thresholds of about 0.1 to 0.3, depending on the dimension type.

The primary usefulness of the histograms has been twofold. First, they have established that there is a statistically significant tendency for failure regions to form clusters. Second, they have provided an indication of which edges in the graph are, in fact, statistically significant. The shortcoming of the histograms is that

they do not show exactly how clusters are being formed. The graphs must actually be constructed for this purpose.

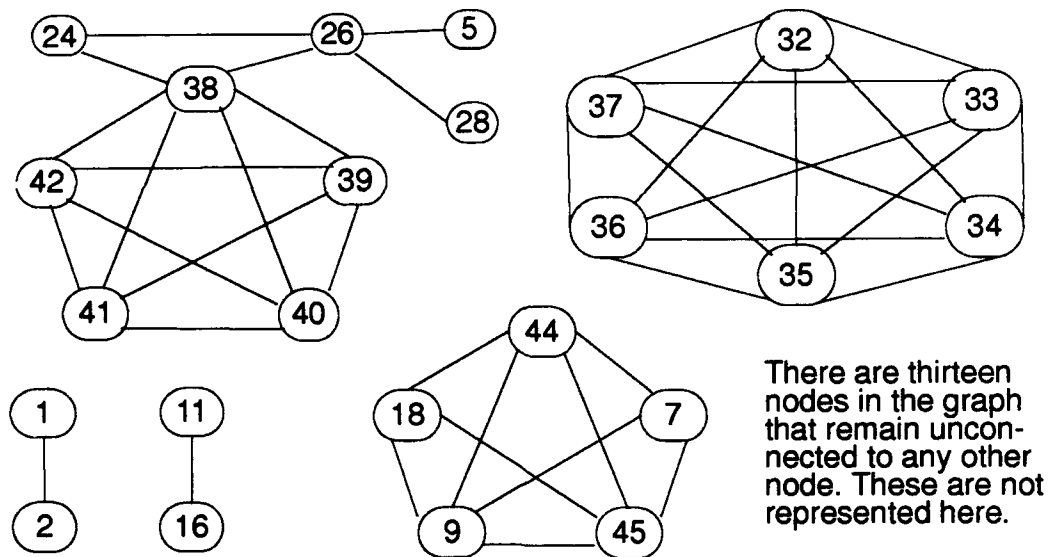
Graphs were constructed using only statistically significant edges. Examples of these graphs are given in Figures 4.1 and 4.2. Most of the graphs are too large to be presented in a graphical format. Complete listings of the edges are presented in Appendix E. The numbering of the nodes is derived from the order in which the faults were discovered in Shimeall and Leveson's study (Shimeall and Leveson, 1991). A complete listing of the numbered faults and their associated failure regions is given in the library of failure regions (Shimeall, 1991).

The Identical dimensions of Versions 1 and 2 displayed behavior similar to that shown in Figure 4.1 for Version 3. Clusters (subgraphs) of two to nine nodes formed with many of the clusters containing components that were complete on three to six nodes. Version 4, on the other hand, had only three two-node clusters above the random level. It is notable that Version 4 displayed the least overall variance from the random regions for the Identical dimension.

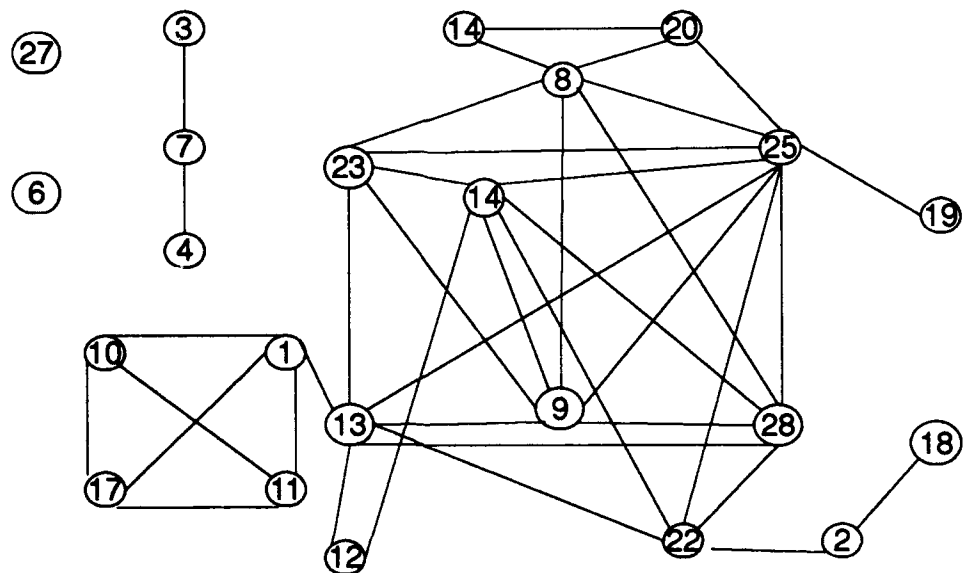
The graphs for the Coincidental dimension tended to be formed in a different fashion. As the example in Figure 4.2 shows, there are several subgraphs that are complete on three or four nodes. However, the clusters are not as clearly separated in the Coincidental dimension as they are in the Identical dimension.

Graphs for the Composite dimension were not constructed since the original purpose of this dimension was simply to ensure that division into Identical and Coincidental did not impose an artificial structure on the data. Review of the table of edges in Appendix E suggests that graphs constructed from the Composite dimension would behave similarly to those of the Coincidental dimension.

The differences in the way the clusters join as the threshold is relaxed suggests that the hierarchical clustering theory may have been appropriate for the Coincidental dimension but that the Identical clustering might, in fact, be better modeled as partitioned.



**Figure 4.1: Version 3 Identical Clusters (coefficient in classes > 0.25)**



**Figure 4.2: Version 1 Coincidental Clusters (coefficient in classes > 0.25)**

#### **D. DATA INTERPRETATION**

Two different clustering behaviors have been noted for the experimental data. The difference in behavior seems to be driven primarily by dimension type. This suggests that failure region clusters may support two different methods of software testing.

The first type of clustering is essentially partitioned, as displayed by the Identical dimensions. This type of clustering would support a testing method that first broadly examines the software, for example all-branches structural testing. Failure regions of faults found by the initial method can then be used to search for other faults in the cluster. Since for partitioned clusters every fault is in exactly one cluster, it would be necessary to find a set of faults that covers several clusters with the initial testing method in order for failure region analysis to be a successful follow-on approach.

The hierarchical clustering exhibited by the Coincidental dimension is more suggestive of an iterative approach to testing. At least one fault must still be found by some other method but it may then be possible to iteratively analyze failure regions and find more faults.

The information available to the tester from failure region analysis is more specific than just which variables should be considered in constructing a test set, especially for the Identical dimension. Failure region analysis gives the tester the specific conditions that resulted in faults. This study has shown that he may reasonably expect these same conditions to appear in the failure region bounds of other faults.

Finally, it should be noted that failure region cluster analysis cannot guarantee that every fault will be located. The primary reason for this is that not every fault will be in a cluster or, more correctly, that some faults may be in singleton clusters. Several of the graphs generated in this study contained singletons. These faults will have to be discovered by some other method.

#### **E. CONCLUSION**

This chapter has presented the results of this experiment and their validity. It has also suggested ways these results may be applied in developing software test-

ing strategies. The last chapter summarizes the findings of this thesis and discusses how these findings support or contrast with previous findings. Directions for further research are also suggested there.

## **V. CONCLUSIONS AND SUGGESTIONS FOR FURTHER RESEARCH**

Previous authors have postulated that faults are related to each other and testers have tried to exploit the effect. However, the evidence and applications have been largely anecdotal. This thesis is the first work that has empirically analyzed the relationships between specific faults by using failure regions. The results of this thesis not only support the existence of such relationships, they suggest methods for explicit rather than just implicit exploitation of them. This chapter summarizes these results, discusses them in the light of previous work, and describes directions for future research that are suggested by this thesis.

### **A. CONCLUSIONS**

This thesis offers strong evidence that failure regions tend to form clusters. The usefulness of this clustering behavior is that known faults in a program can be analyzed to produce their failure regions. Those failure regions then provide information about variables and conditions that are likely to be involved in other failure regions. This, in turn, suggests to the tester areas that will probably be fruitful in his search for other faults.

Failure region clustering was observed based on two distinct criteria: shared bounding conditions (the Identical dimension) and shared variables that appear in different contexts (the Coincidental dimension). The nature of the cluster formation for the two dimensions, however, was markedly different. The Identical dimension tended to produce small, isolated, strongly connected clusters. The nodes in these clusters were defined at relatively high thresholds and then the clusters became more strongly connected as the edge weight threshold was lowered. On the other hand, the Coincidental dimension tended to form larger, less strongly connected clusters. The clusters that formed at higher thresholds tended to merge into one or two larger clusters as the edge weight threshold was relaxed. There was no strongly identifiable pattern to the Coincidental cluster formation.

## **B. RELATIONSHIP OF RESULTS TO PREVIOUS WORK**

The results of this thesis generally support the findings of previous researchers in the area of relationships between faults. This agreement gives some confidence that these results may extend to more general software applications. The results also offer some amplification to previous studies.

Offutt offered convincing empirical evidence that the coupling effect existed, but his study provided no explanation of the basis for it (Offutt, 1989). This thesis makes the first step toward identifying the specific behaviors of faults that result in this effect. The Identical dimensions considered in this study closely resemble the idea behind mutation testing: change one condition at a time and run a new test. Thus, the small, strongly connected, isolated clusters that were formed in the Identical dimension graphs provide an explanation of why multiple-mutation testing does not fair significantly better than single-mutation testing. Every fault in the cluster has a short path to most other faults in the cluster.

The Identical dimension results also seem to support Hamlet and Taylor's analysis that partitioned testing is a good debugging method but a poor technique for release testing (Hamlet and Taylor, 1988). If one fault in a cluster is known, its failure region may allow the partitions to be refined in a way that leads to the other faults in the cluster. However, this does not aid in finding faults in other clusters and, in general, failure region analysis offers no confidence that every cluster of faults has been located.

The graphs formed in the Coincidental dimension are more eccentric and suggest a complex behavior that is more difficult to analyze than the Identical dimension graphs. The absence of a clearly evolving structure in the Coincidental graph formation may offer insight into when specific testing techniques are appropriate. These graphs would seem to provide a basis for Hamlet and Taylor's assertion that in the absence of specific information that allows technique refinement (such as that provided by the Identical dimension), random testing is as reliable as the best planned testing (Hamlet and Taylor, 1988). Further study of the Coincidental dimension is needed to clarify its implications for software testing.

## **C. SUGGESTIONS FOR FURTHER RESEARCH**

### **1. Experimental Method**

While the results of this work are promising, the experimental population was small and narrowly focused. Additionally, the programs were written by students. Both the method and the results should be validated using a broad range of professionally produced applications.

One weakness of the method used in this thesis is in how it deals with one failure region that is a subset of another. For instance, if region 1 is bounded by conditions B and C and region 2 is bounded by A, B, C, D, E and F, the Jaccard coefficient is 0.33. However, the relationship is probably stronger than is suggested by the coefficient.

Another weakness is inherent in the use of a coefficient for weighting the edges of the graphs. The relationships described by the coefficient are actually rational rather than real. The ratios  $1/2$  and  $6/12$  both yield the same coefficient. However, the second ratio probably represents a more involved (and perhaps more easily exploitable) relationship. Both of these difficulties with the coefficient suggest the need for a more descriptive representation of the relationship between failure regions. The separate distributions of the numerators and the denominators were used in an initial attempt to exploit this difference in ratios. However, this approach offered no insight and was omitted from this thesis.

Finally, only threshold graphs were used for cluster analysis. Alternative approaches to the problem were described in Chapters II and III. These and other methods should be explored. Analysis for k-connected components seems particularly promising in light of the graphs presented in the previous chapter.

### **2. Related Questions**

Several questions about fault and failure region behavior have arisen from this study. First, several failure regions were identical in one condition, e.g., the reachability condition, but differed in the other two conditions. Can the method developed in this study (or some other method) take advantage of this special



behavior? What if Condition I for one failure region is identical to Condition II or III for other failure regions?

Somewhat different behaviors were noted among the three versions that had approximately the same number of failure regions. Is there an antagonistic affect between the Coincidental and Identical dimensions? Does strong clustering in one dimension mean weak clustering in the other?

Faults were numbered in order as they were discovered by the various testing techniques of Shimeall and Leveson's study (Shimeall and Leveson, 1991). Thus, many sequentially numbered faults were discovered by the same fault-detection method. Many sequentially numbered faults were also strongly connected in the graphs, often at the same threshold values. Is there an identifiable relationship between certain fault-detection techniques and certain types of fault clusters?

An in-depth study of the clusters identified in this study may be useful in determining specifically which types of conditions and variables are most likely to cause clusters to form. This is an area where comparison of different software applications is especially important. Even if clustering is a characteristic of failure regions in general, the specific types of conditions that cause the clustering may vary from application to application.

Finally, the understanding of relationships between faults that this thesis offers may provide insight into refining existing fault-detection techniques. For example, all-paths testing is generally considered to be a desirable goal; however, it is usually not achievable because the number of paths is too large. The key relationships identified by failure region analysis may provide the information necessary to be able to modify such techniques so that they are practical.

#### **D. APPLICATIONS BEYOND TESTING**

This study has focused on the use of failure regions to understand the relationship of one fault to another. The goal has been to develop information that will be useful in software testing. In a broader sense, however, the relationship between two failure regions is a condensation of the relationships between the two sets of

code locations that are associated with those failure regions. A set of bounding conditions that is analogous to a failure region can be developed for any location in a program. If the conclusions of this thesis are applied from this perspective, it may lead to a better understanding of how different parts of a program interact with each other. Such an understanding might help prevent occurrences like the telephone example cited in the first chapter by allowing failure prediction.

## APPENDIX A

### GRAPH THEORY DEFINITIONS

The following definitions are taken from Buckley and Harary (Buckley and Harary, 1990):

1. A **graph** consists of a finite nonempty set  $N$  of nodes together with a set  $E$  of edges. An **edge** is an unordered pair of distinct nodes in  $N$ .
2. A **path** from node  $u$  to node  $v$  is a sequence of distinct nodes and edges that starts with  $u$  and ends with  $v$ . The **length** of a path is equal to the number of edges in the path.
3. The **distance** between nodes  $u$  and  $v$  is equal to the length of a shortest  $u$ - $v$  path.
4. A graph is **connected** if there is a path joining each pair of nodes.
5. A **component** of a graph is a maximal connected subgraph.
6. A **cutnode (bridge)** of a graph is a node (edge) whose removal increases the number of components.
7. A **nonseparable** graph is connected, nontrivial, and has no cutnodes.
8. A **block** of a graph is a maximal nonseparable subgraph.
9. The **eccentricity** of a node  $v$  is the distance to a node farthest from  $v$ .
10. The **radius (diameter)** of a graph is the minimum (maximum) eccentricity of all nodes in the graph.
11. The **center** of a graph is the set of all nodes whose eccentricity equals the radius of the graph.
12. The **connectivity (edge-connectivity)** of a graph is the minimum number of nodes (edges) whose removal results in a disconnected or trivial graph.
13. A graph is  **$n$ -connected** ( $n$ -edge connected) if its connectivity (edge-connectivity) is at least  $n$ .

## APPENDIX B

### VERSION 1 FAILURE REGIONS

#### Notation

In the descriptions that follow, the following conventions are used:

- So far as is possible, the conventions of the specification have been preserved.
- Text appearing in italics (e.g. '*Endurance*') are defined within the scope of this document, either globally or for a specific failure region.
- Text appearing in roman type (e.g. '*Army*['*Endurance*']') are program variables for the implementations containing the fault. The only exception to this is the variable '*Mainloop*', which is used to indicate the current simulation cycle, but may not appear in a specific version under that name.
- Due to the fact that program variables are more than one character in length, all multiplication is shown explicitly with the multiplication symbol  $\times$ .
- Due to the length of the formulae below, it is necessary to break formulae across more than one line. There are no matrix or vector operations appearing in this document, and parentheses are used strictly to delimit portions of formulae to improve readability or to indicate precedence of operations.
- All definitions within 'Condition I' of a failure region are assumed to extend over 'Condition II' and 'Condition III' of that failure region unless use of parentheses indicates otherwise. All definitions within 'Condition II' of a failure region are similarly assumed to extend over 'Condition III' of that failure region.
- The diacritical marks ' and " are used strictly to distinguish between variables of similar name and role in a given failure region.

## Predicate Definitions

Endurance of Squadron ( $B, g, j$ ) at time  $t$ :

$$Endurance(B, g, j, t) = Army[B, g].Endurance[j] - \\ Army[B, g].Wear[j] \times t - \\ Damage(B, g, j, t-1) + Repair(B, g, j, t-1)$$

Weapon Damage of Squadron( $B, g, j$ ) up to and including time  $t$ :

$$Damage(B, g, j, t) = \begin{cases} 0 & \text{if } t \leq 1 \\ Damage(B, g, j, t-1) + \\ N_{Army[\neg B]} \left( \sum_{e=1}^{Params.NumWTypes} \left( \sum_{w=1}^{Army[\neg B, e].Weapon[w].NumWeapon} \sum_{i=1}^{Army[\neg B, e].Weapon[w].Damagex \times} \right. \right. \\ \left. \left. \begin{aligned} & Army[\neg B, g].WeapSensativity[w] \times \\ & \max \left( 0, 1 - \frac{\sqrt{(x_{B, g, j}(t-1) - ax_{\neg B, e, w, i}(t-1))^2 + (y_{B, g, j}(t-1) - ay_{\neg B, e, w, i}(t-1))^2}}{Army[\neg B, e].Weapon[w].Radius} \right) \right) \right) \end{aligned} \right) & \text{otherwise} \end{cases}$$

Whether or not Squadron( $B, g, j$ ) is a casualty at time  $t$ :

$$Casualty(B, g, j, t) \equiv (Endurance(B, g, j, t-1) > 0) \wedge \\ \left( \frac{Endurance(B, g, j, t-1)}{Army[B, g].Endurance[j]} \leq 0.5 \right)$$

Repair applied to Squadron( $B, g, j$ ) up to and including time  $t$ :

$$Repair(B, g, j, t) = \begin{cases} 0 & \text{if } t \leq 1 \\ Repair(B, g, j, t-1) & \text{if } (t > 1) \wedge \neg Casualty(B, g, j, t-1) \\ Repair(B, g, j, t-1) + \\ \min(Suppl(B, g, t-1)/NumCas(B, g, t), \\ FixRate(B, g, t-1)/NumCas(B, g, t), \\ (Army[B, g].Endurance[j] \\ - Endurance(B, g, j, t-1) \\ - Repair(B, g, j, t-1) \\ + Repair(B, g, j, t-2))) & \text{otherwise} \end{cases}$$

Number of Casualties in Battalion  $B, g$  at time  $t$ :

$$NumCas(B, g, t) = \sum_{j=1}^{Army[B, g].Squadrons} \begin{cases} 1 & \text{if } Casualty(B, g, j, t-1) \\ 0 & \text{otherwise} \end{cases}$$

Rate of Repair available to any squadron of battalion  $B, g$  at time  $t$ :

$$FixRate(B, g, t) = Army[B, g].FixRate \times NumFix(B, g, t-1)$$

Number of Squadrons in battalion  $B, g$  dedicated to repair other squadrons at time  $t$ :

$$NumFix(B, g, t) = Army[B, g].NumFixers \times \frac{\sum_{j=1}^{Army[B, g].Squadrons} \begin{cases} 0 & \text{if } \neg Casualty(B, g, j, t) \\ 1 & \text{otherwise} \end{cases}}{Army[B, g].Squadrons}$$

Amount of supplies available in battalion  $B, g$  at time  $t$ :

$$Suppl(B, g, t) = Army[B, g].FixSuppl - \sum_{j=1}^{Army[B, g].Squadrons} Repair(B, g, j, t-1)$$

X Location of Battalion  $B, g$  at time  $t$ :

$$\begin{aligned} x_{B,g}(t) = & Army[B, g].X + \\ & \sum_{d=1}^t (V(B, g, d) \times \cos(Army[B, g].Theta) \\ & \times TM(B, g, x_{B,g}(d-1), y_{B,g}(d-1), V(B, g, d-1)) \\ & \times WM(B, g, x_{B,g}(d-1), y_{B,g}(d-1), d)) \end{aligned}$$

Y Location of Battalion  $B, g$  at time  $t$ :

$$\begin{aligned} y_{B,g}(t) = & Army[B, g].Y + \\ & \sum_{d=1}^t (V(B, g, d) \times \sin(Army[B, g].Theta) \\ & \times TM(B, g, x_{B,g}(d-1), y_{B,g}(d-1), V(B, g, d-1)) \\ & \times WM(B, g, x_{B,g}(d-1), y_{B,g}(d-1), d)) \end{aligned}$$

Velocity of Battalion  $B, g$  at time  $t$ :

$$V(B, g, t) = \min_{j=1}^{Army[B, g].Squadrons} \begin{cases} \infty & \text{if } Endurance(B, g, j, t-1) \leq 0 \\ Army[B, g].V0[j] \times \frac{Endurance(B, g, j, t-1)}{Army[B, g].Endurance[j]} & \text{otherwise} \end{cases}$$

Terrain effect on Movement of Battalion  $B, g$  at location  $x, y$  moving at velocity  $v$ :

Let  $x'$  and  $y'$  represent the end of the possible movement,  $p, q$  be the Terrain grid location of  $x, y$ :

$$x' = x + v \times \cos(\text{Army}[B, g].\text{Theta})$$

$$y' = y + v \times \sin(\text{Army}[B, g].\text{Theta})$$

$$p(x) = \lfloor \frac{x}{\text{Params.XDelta}} \rfloor$$

$$q(y) = \lfloor \frac{y}{\text{Params.YDelta}} \rfloor$$

$$TM(B, g, x, y, v) = \begin{cases} 0 & \text{if } v = 0 \\ \max \left( 0, \frac{\text{Army}[B, g].\text{MaxSlope} - \frac{\text{Alt}(p(x'), q(y'), x', y') - \text{Alt}(p(x), q(y), x, y)}{\sqrt{(x'-x)^2 + (y'-y)^2}}}{\text{Army}[B, g].\text{MaxSlope}} \right) & \text{otherwise} \end{cases}$$

Weather effect on Movement of Battalion  $B, g$  at location  $x, y$  at time  $t$ :

Let  $(WX_i, WY_i)$  be the center location of storm  $i$  at time  $t$ :

$$WX_i = \begin{cases} \text{Weather}[i].WX0 & \text{if } t < \text{Weather}[i].TStart \vee t > \text{Weather}[i].TEnd \\ \text{Weather}[i].WX0 + (t - \text{Weather}[i].TStart) \times \text{Weather}[i].dWX & \text{otherwise} \end{cases}$$

$$WY_i = \begin{cases} \text{Weather}[i].WY0 & \text{if } t < \text{Weather}[i].TStart \vee t > \text{Weather}[i].TEnd \\ \text{Weather}[i].WY0 + (t - \text{Weather}[i].TStart) \times \text{Weather}[i].dWY & \text{otherwise} \end{cases}$$

Let  $W$  be the total effect of storms on location  $(x, y)$  at time  $t$ :

$$W(x, y, t) = \sum_{i=1}^{\text{Params.NumWEvents}} \begin{cases} 0 & \text{if } t < \text{Weather}[i].TStart \vee t > \text{Weather}[i].TEnd \\ \max \left( 0, \frac{\text{Weather}[i].WRadius - \sqrt{(x - WX_i)^2 + (y - WY_i)^2}}{\text{Weather}[i].WRadius} \times \text{Weather}[i].WSeverity \right) & \text{otherwise} \end{cases}$$

$$WM(B, g, x, y, t) = \begin{cases} 1 & \text{if } W(x, y, t) = 0 \\ \text{Army}[B, g].MWEff \times \left| \frac{W(x, y, t) - \text{Params.WMaxSeverity} \times \text{Params.NumWEvents}}{\text{Params.WMaxSeverity} \times \text{Params.NumWEvents}} \right| & \text{otherwise} \end{cases}$$

Weather effect on Observation at location  $(x, y)$  at time  $t$ :

$$WO(x, y, t) = \begin{cases} 0 & \text{if } W(x, y, t) = 0 \\ \left\lfloor \frac{W(x, y, t) - \text{Params.WMaxSeverity} \times \text{Params.NumWEvents}}{\text{Params.WMaxSeverity} \times \text{Params.NumWEvents}} \right\rfloor & \text{otherwise} \end{cases}$$

$(X, Y)$  Location of Squadron  $B, g, j$  at time  $t$ :

Let  $s$  be the number of Squadrons in Battalion  $B, g$  prior to squadron  $j$  that have positive endurance at time  $t$ :

$$s(B, g, j, t) = \sum_{i=1}^{j-1} \begin{cases} 0 & \text{if } \text{Endurance}(B, g, i, t-1) \leq 0 \\ 1 & \text{otherwise} \end{cases}$$

$$x_{B,g,j}(t) = \begin{cases} x_{B,g}(t-1) + \text{Army}[B, g].\text{SquadSep} \times \left( s(B, g, j, t) - \left\lfloor \frac{s(B, g, j, t)}{\text{Army}[B, g].\text{GRow}} \right\rfloor \times \text{Army}[B, g].\text{GRow} \right) - \frac{\text{Army}[B, g].\text{GRow} \times \text{Army}[B, g].\text{SquadSep}}{2} & \text{if } s(B, g, \text{Army}[B, g].\text{Squadron} + 1, t) - s(B, g, j, t) > \text{Army}[B, g].\text{GRow} \\ x_{B,g}(t-1) + \text{Army}[B, g].\text{SquadSep} \times \left( s(B, g, j, t) - \left\lfloor \frac{s(B, g, j, t)}{\text{Army}[B, g].\text{GRow}} \right\rfloor \times \text{Army}[B, g].\text{GRow} \right) - \frac{s(B, g, \text{Army}[B, g].\text{Squadron} + 1, t) - \left\lfloor \frac{s(B, g, \text{Army}[B, g].\text{Squadron} + 1, t)}{\text{Army}[B, g].\text{GRow}} \right\rfloor \times \text{Army}[B, g].\text{GRow}}{2} & \\ \times \text{Army}[B, g].\text{SquadSep} & \text{otherwise} \end{cases}$$

$$y_{B,g,j}(t) = y_{B,g}(t-1) + \text{Army}[B, g].\text{RowSep} \times \left\lfloor \frac{s(B, g, j, t)}{\text{Army}[B, g].\text{GRow}} \right\rfloor - 0.5 \times \left\lfloor \frac{s(B, g, \text{Army}[B, g].\text{Squadron} + 1, t)}{\text{Army}[B, g].\text{GRow}} \right\rfloor \times \text{Army}[B, g].\text{RowSep}$$



Squadron  $B, g, j$  observes squadron  $\neg B, e, k$  at time  $t$ :

$$\text{Observe}(B, g, j, e, k, t) \equiv \text{BigEnough}(B, g, j, e, k, t) \wedge \text{Clear}(B, g, j, e, k, t) \\ \wedge \text{Obvious}(B, g, j, e, k, t)$$

Squadron  $\neg B, e, k$  is large enough to be seen at the distance from squadron  $B, g, j$  at time  $t$ :

$$\text{BigEnough}(B, g, j, e, k, t) \equiv \\ xgj = x_{B,g,j}(t-1) \wedge ygj = y_{B,g,j}(t-1) \wedge \\ xek = x_{\neg B,e,k}(t-1) \wedge yek = y_{\neg B,e,k}(t-1) \wedge \\ \left( \max \left\{ \tan^{-1} \left( \frac{y'-y}{x'-x} \right) - \tan^{-1} \left( \frac{y''-y}{x''-x} \right) \right\} \right. \\ \left. (x', y'), (x'', y'') \in \{(xek \pm \text{Army}[\neg B, e].\text{SquadWidth}/2, \right. \\ \left. yek \pm \text{Army}[\neg B, e].\text{SquadLength}/2)\} \right. \\ \left. \geq \text{Army}[B, g].\text{ObsMinAngle}[j] \right)$$

No terrain blocks the view of squadron  $\neg B, e, k$  from the position of squadron  $B, g, j$  at time  $t$ :

$$\text{Clear}(B, g, j, e, k, t) \equiv \\ xgj = x_{B,g,j}(t-1) \wedge ygj = y_{B,g,j}(t-1) \wedge \\ xek = x_{\neg B,e,k}(t-1) \wedge yek = y_{\neg B,e,k}(t-1) \wedge \\ (\forall a, a', c, c', z, z', a = \lfloor \frac{xgj}{\text{Params.XDelta}} \rfloor \wedge a' = \lfloor \frac{xek}{\text{Params.XDelta}} \rfloor \wedge \\ c = \lfloor \frac{ygj}{\text{Params.YDelta}} \rfloor \wedge c' = \lfloor \frac{yek}{\text{Params.YDelta}} \rfloor \wedge \\ z = \text{Alt}(a, c, xgj, ygj) \wedge z' = \text{Alt}(a', c', xek, yek) \wedge \\ (\forall n, 1 \leq n < \text{Params.SampleRate} - 1, \\ (\exists r, p, q, r = \frac{n}{\text{Params.SampleRate} - 1}, p = \lfloor \frac{xgj + r \times (xek - xgj)}{\text{Params.XDelta}} \rfloor, q = \lfloor \frac{ygj + r \times (yek - ygj)}{\text{Params.YDelta}} \rfloor, \\ (z + r \times (z' - z)) > \text{Alt}(p, q, xgj + r \times (xek - xgj), ygj + r \times (yek - ygj)) \\ ) ) )$$

Squadron  $\neg B, e, k$  differs enough from its background to be discerned by squadron  $B, g, j$  at time  $t$ :

$$\begin{aligned}
 & \text{Obvious}(B, g, j, e, k, t) \equiv \\
 & \quad xgj = x_{B,g,j}(t-1) \wedge ygj = y_{B,g,j}(t-1) \wedge \\
 & \quad xek = x_{\neg B,e,k}(t-1) \wedge yek = y_{\neg B,e,k}(t-1) \wedge \\
 & \quad \left( \left( \frac{BI(a', e', xek, yek) - \text{Army}[\neg B, e].\text{SquadIntensity}[k]}{BI(a', e', xek, yek)} - \right. \right. \\
 & \quad \sum_{n=1}^{\text{Params.SampleRate}} \left( \left( WO \left( xgj \times \frac{n \times (xek - xgj)}{\text{Params.SampleRate}}, ygj \times \frac{n \times (yek - ygj)}{\text{Params.SampleRate}}, \text{Mainloop} \right) \times \right. \right. \\
 & \quad \quad \left. \left. \text{Army}[B, g].\text{VWEffEffect} \right) + \right. \\
 & \quad \left. \sum_{e'=1}^{N\text{Army}[\neg B]} \left\{ \begin{array}{ll} 0 & \text{if } \sqrt{\left( x_{\neg B,e'}(\text{Mainloop} - 1) - xgj \times \frac{n \times (xek - xgj)}{\text{Params.SampleRate}} \right)^2 + \left( y_{\neg B,e'}(\text{Mainloop} - 1) - ygj \times \frac{n \times (yek - ygj)}{\text{Params.SampleRate}} \right)^2} \right. \\ & \quad \left. > \text{Army}[\neg B, e']. \text{ObsJamRadius} \right. \\ & \quad \left. \sqrt{\frac{\left( x_{\neg B,e'}(\text{Mainloop} - 1) - xgj \times \frac{n \times (xek - xgj)}{\text{Params.SampleRate}} \right)^2 + \left( y_{\neg B,e'}(\text{Mainloop} - 1) - ygj \times \frac{n \times (yek - ygj)}{\text{Params.SampleRate}} \right)^2}{\text{Army}[\neg B, e']. \text{ObsJamRadius}}} \right. \\ & \quad \left. \times \text{Army}[\neg B, e']. \text{ObsJamEffect} \text{ otherwise} \right. \\ & \quad \left. \left. \right) \right) < \text{Army}[B, g]. \text{ObsMinContrast}[j] \Big)
 \end{aligned}$$

Squadron  $\neg B, e, k$  is in range of the weapons of battalion  $B, g$  at time  $t$ :

$$\begin{aligned}
 & \text{InRange}(B, g, e, k, t) \equiv \\
 & \quad xek = x_{\neg B,e,k}(t-1) \wedge yek = y_{\neg B,e,k}(t-1) \wedge \\
 & \quad \sqrt{(xek - x_{B,g}(t-1))^2 + (yek - y_{B,g}(t-1))^2} < \text{Army}[B, g]. \text{Weapon}[i]. \text{Range}
 \end{aligned}$$

Number of Squadrons in battalion  $B, g$  dedicated to processing messages at time  $t$ :

$$\text{NumProcess}(B, g, t) = \text{Army}[B, g]. \text{NumProcess} \times \frac{\text{NumCas}(B, g, t)}{\text{Army}[B, g]. \text{Squadrons}}$$

Number of Squadrons in battalion  $B, g$  dedicated to receiving messages at time  $t$ :

$$\text{NumRec}(B, g, t) = \text{Army}[B, g]. \text{NumReceive} \times \frac{\text{NumCas}(B, g, t)}{\text{Army}[B, g]. \text{Squadrons}}$$

Number of Squadrons in battalion  $B, g$  dedicated to communications jamming at time  $t$ :

$$\text{NumJam}(B, g, t) = \text{Army}[B, g]. \text{NumJammers} \times \frac{\text{NumCas}(B, g, t)}{\text{Army}[B, g]. \text{Squadrons}}$$

Number of functional weapons of type  $i$  in battalion  $B, g$  at time  $t$ :

$$\text{NumWeapon}(B, g, i, t) = \text{Army}[B, g]. \text{Weapon}[i]. \text{NumWeapon} \times \frac{\text{NumCas}(B, g, t)}{\text{Army}[B, g]. \text{Squadrons}}$$

Target coordinates for weapon  $i$  of type  $w$  in Battalion  $B, g$  at time  $t$ :

$$\begin{aligned}
 ax_{B,g,w,i}(t) &= x_{\neg B,e,k} \ni \\
 &\left( \left( \sum_{e'=1}^{e-1} \sum_{k'=1}^{\text{Army}} [\neg B, e']. \text{Squadrons} \begin{cases} 1 & \text{if } \exists j, \text{Observe}(B, g, j, e', k', t-1) \\ 0 & \text{otherwise} \end{cases} \right) \right. \\
 &+ \left. \left( \sum_{k'=1}^{k-1} \begin{cases} 1 & \text{if } \exists j, \text{Observe}(B, g, j, e, k', t-1) \\ 0 & \text{otherwise} \end{cases} \right) \right) = \\
 &\left( \sum_{w'=1}^{w-1} \text{NumWeapon}(B, g, w', t) \right) + i - 1
 \end{aligned}$$

$$\begin{aligned}
 ay_{B,g,w,i}(t) &= y_{\neg B,e,k} \ni \\
 &\left( \left( \sum_{e'=1}^{e-1} \sum_{k'=1}^{\text{Army}} [\neg B, e']. \text{Squadrons} \begin{cases} 1 & \text{if } \exists j, \text{Observe}(B, g, j, e', k', t-1) \\ 0 & \text{otherwise} \end{cases} \right) \right. \\
 &+ \left. \left( \sum_{k'=1}^{k-1} \begin{cases} 1 & \text{if } \exists j, \text{Observe}(B, g, j, e, k', t-1) \\ 0 & \text{otherwise} \end{cases} \right) \right) = \\
 &\left( \sum_{w'=1}^{w-1} \text{NumWeapon}(B, g, w', t) \right) + i - 1
 \end{aligned}$$

Command Message  $m$  Implemented in Battalion  $B, g$  before time  $t$ :

$$\begin{aligned}
 \text{Mimp}(B, g, m, t) &\equiv \\
 &((\text{Cmsgs}[B, m].\text{Time} + \text{Army}[B, g].\text{MediaDelay} \\
 &+ \text{RecDelay}(B, g, \text{RecT}(B, g, m)) + \text{QueDelay}(B, g, m) \\
 &+ \text{Army}[B, g].\text{ProcDelay}) < t) \wedge \\
 &(\text{Cmsgs}[B, m].\text{Dest} = g)
 \end{aligned}$$

Delay due to message receipt at battalion  $B, g$  at time  $t$ :

$$\text{RecDelay}(B, g, t) = \begin{cases} \infty & \text{if } \text{NumRec}(B, g, t) - \text{ComJam}(B, g, t) \leq 0 \\ \frac{\text{Army}[B, g].\text{RecRate}}{\text{NumRec}(B, g, t) - \text{ComJam}(B, g, t)} & \text{otherwise} \end{cases}$$

Number of jammed receivers in battalion  $B, g$  at time  $t$ :

$$\begin{aligned}
 \text{ComJam}(B, g, t) &= \\
 \text{NArmY}[\neg B] &\sum_{e=1} \min(\text{NumJam}(\neg B, e, t), \text{Army}[\neg B, e].\text{CommJamPriority}[g]) \times \\
 &\text{Army}[\neg B, e].\text{CommJamEff} \times \\
 &\max \left( 0, \frac{\text{Army}[\neg B, e].\text{CommJamRadius} - \sqrt{(x_{\neg B,e}(t-1) - x_{B,g}(t-1))^2 + (y_{\neg B,e}(t-1) - y_{B,g}(t-1))^2}}{\text{Army}[\neg B, e].\text{CommJamRadius}} \right)
 \end{aligned}$$

Delay due to message queuing of command message  $m$  in Battalion  $B, g$ :

$$\text{QueDelay}(B, g, m) = \sum_{t=\text{RecT}(B, g, m)}^{\text{Duration}} \begin{cases} 1 & \text{if } \text{CmdSum}(B, g, m, t) + \text{ReptSum}(B, g, m, t) \\ & \geq \text{NumProcess}(B, g, t-1) \\ 0 & \text{otherwise} \end{cases}$$

Time command message  $m$  is received at battalion  $B, g$ :

$$\text{RecT}(B, g, m) = \text{Cmsgs}[B, m].\text{Time} + \text{Army}[B, g].\text{MediaDelay}$$

Time delay for report message from battalion  $B, f$  to be transmitted to battalion  $B, g$ :

$$\text{RepT}(B, g, f) = \text{Army}[B, f].\text{SendRate} + \text{Army}[B, g].\text{MediaDelay}$$

Number of command messages, other than  $m$  being processed by battalion  $B, g$  at time  $t$ :

$$CmdSum(B, g, m, t) = \sum_{n=1}^{NCmsgs[B]} \begin{cases} 0 & \text{if } (m = n) \vee (Cmsgs[B, n].Dest \neq g) \vee \\ & (t \leq RecT(B, g, n) \wedge \\ & Cmsgs[B, m].Priority > Cmsgs[B, n].Priority) \\ & \vee (Cmsgs[B, n].Time > t) \vee \\ & (RecT(B, g, n) + Army[B, g].ProcDelay < t) \\ 1 & \text{otherwise} \end{cases}$$

Some opposing squadron exists and is observed by a squadron of  $B, g$ , at time  $t$ :

$$SomeObserve(B, g, t) \equiv (\exists e, 1 \leq e \leq NArmy[\neg B], Army[\neg B, e].Squadrons > 0 \wedge EObserve(B, g, e, t))$$

Some opposing squadron in battalion  $\neg B, e$ , exists and is observed by some squadron of  $B, g$  at time  $t$ .

$$EObserve(B, g, e, t) \equiv (\exists k, 1 \leq k \leq Army[\neg B, e].Squadrons, Endurance(\neg B, e, k, t) > 0 \wedge (\exists j, 1 \leq j \leq Army[B, f].Squadrons, Endurance(B, f, j, t) > 0 \wedge Observe(B, g, j, e, k, t)))$$

Number of report messages being processed by battalion  $B, g$  at time  $t$ , while message  $m$  may be queued:

$$ReptSum(B, g, m, t) = \sum_{f=1}^{NArmy[B]} \begin{cases} 0 & \text{if } (Army[B, f].Report \neq g) \vee \\ & (\forall t', t - RepT(B, g, f) - Army[B, g].ProcDelay \\ & \leq t' \leq t - RepT(B, g, f), \\ & \neg SomeObserve(B, f, t')) \vee \\ & (SomeObserve(B, f, t - RepT(B, g, f)) \wedge \\ & Army[B, f].Priority < Cmsgs[B, m].Priority) \\ 1 & \text{otherwise} \end{cases}$$

Battalion  $B, g$  is active:

$$Active(B, g) \equiv ((Duration > 0) \wedge (Mainloop \in \{0 \dots Duration\}) \wedge (B \in \{TRUE, FALSE\}) \wedge (NArmy[B] > 0) \wedge (g \in \{1 \dots NArmy[B]\}) \wedge (Army[B, g].Squadrons > 0) \wedge (\exists i, 1 \leq i \leq Army[B, g].Squadrons, Endurance(B, g, i, Mainloop) > 0))$$

Altitude at position  $(x, y)$  in Terrain grid  $(p, q)$ :

$$Alt(p, q, x, y) = \left( \frac{Terrain[p, q] - Terrain[p + 1, q] - Terrain[p, q + 1] + Terrain[p + 1, q + 1]}{Params.XDelta \times Params.YDelta} \times x \times y \right) + \left( \frac{q(Terrain[p, q + 1] - Terrain[p + 1, q + 1]) - (q + 1)(Terrain[p, q] - Terrain[p + 1, q])}{Params.XDelta} \times x \right) + \left( \frac{p(Terrain[p + 1, q] - Terrain[p + 1, q + 1]) - (p + 1)(Terrain[p, q] - Terrain[p, q + 1])}{Params.YDelta} \times y \right) + (p + 1)((q + 1)Terrain[p, q] - qTerrain[p, q + 1]) - p((q + 1)Terrain[p + 1, q] - qTerrain[p + 1, q + 1])$$

Background Intensity at position  $(x, y)$  in Terrain grid  $(p, q)$ :

$$\begin{aligned}
 BI(p, q, x, y) = & \sqrt{\left( \frac{\text{Terrain}[p, q+1] - \text{Terrain}[p, q] + \text{Terrain}[p+1, q+1] - \text{Terrain}[p+1, q]}{2(\text{Params.XDelta})} \right)^2 + \left( \frac{\text{Terrain}[p+1, q+1] - \text{Terrain}[p, q+1] + \text{Terrain}[p+1, q] - \text{Terrain}[p, q]}{2(\text{Params.YDelta})} \right)^2} \\
 & \times \text{Params.ISlopeFactor} + \\
 & \text{Params.IAltFactor} \frac{\text{Params.IMeanAlt} - \text{Alt}(p, q, x, y)}{\text{Params.IMeanAlt}} + \\
 & \text{Params.IX} \times x + \text{Params.IY} \times y + \text{Params.IC}
 \end{aligned}$$

## Failure Region Definitions

### 1.1: Incorrect handling of NumCas when Army.Squadrons=0 initially

Condition I:

$$\text{Duration} > 0 \wedge (\exists B, B \in \{\text{true}, \text{false}\}, \text{NArmy}[B] > 0$$

Condition II:

$$(\exists g, 1 \leq g \leq \text{NArmy}[B], \text{Army}[B, g].\text{Squadrons} = 0))$$

Condition III: True

### 1.2: Update always implements commands ready at the same time in CMsgs array order

Condition I:

$$\begin{aligned} & \text{Active}(B, g) \wedge \\ & (\exists m, n, 1 \leq m \leq \text{NCmsgs}[B], 1 \leq n \leq \text{NCmsgs}[B], m < n \wedge \\ & \quad \text{Mimp}(B, g, m, \text{Mainloop}) \wedge \neg \text{Mimp}(B, g, m, \text{Mainloop} - 1) \wedge \\ & \quad \text{Mimp}(B, g, n, \text{Mainloop}) \wedge \neg \text{Mimp}(B, g, n, \text{Mainloop} - 1)) \end{aligned}$$

Condition II:

$$\text{Cmsgs}[B, m].\text{Priority} < \text{Cmsgs}[B, n].\text{Priority} \wedge \text{Cmsgs}[B, m].\text{msg} \neq \text{Cmsgs}[B, n].\text{msg}$$

Condition III:

$$\begin{aligned} & (\exists i, 1 \leq i \leq \text{NCmsgs}[B], i \neq m \wedge i \neq n \wedge \\ & \quad \text{Mimp}(B, g, i, \text{Duration}) \wedge \neg \text{Mimp}(B, g, i, \text{Mainloop} - 1)) \end{aligned}$$

### 1.3: Over-restrictive check: positive dWX

Condition I: Params.NumWEvents > 0

Condition II:

$$\exists i, 1 \leq i \leq \text{Params.NumWEvents}, \text{Weather}[i].\text{dWX} < 0$$

Condition III: True

**1.4: Over-restrictive check: positive dWY****Condition I:**Params.NumWEvents > 0**Condition II:**

$$\exists i, 1 \leq i \leq \text{Params.NumWEvents}, \text{Weather}[i].\text{dWY} < 0$$

**Condition III:**True**1.5: Garbage value in FixSuppl when Fix Supplies exhausted****Condition I:**

$$\text{Active}(B, g) \wedge (\exists j, 1 \leq j \leq \text{Army}[B, g].\text{Squadrons}, \text{Casualty}(B, g, j, \text{Mainloop}))$$

**Condition II:**

$$\left( \sum_{i=1}^{\text{Army}[B, g].\text{Squadrons}} \text{Repair}(B, g, i, \text{Mainloop}) \right) \geq \text{Army}[B, g].\text{FixSuppl}$$

**Condition III:**

$$(\nexists i, 1 \leq i \leq \text{NCmsgs}[B], \text{Mimp}(B, g, i, \text{Duration}) \wedge \neg \text{Mimp}(B, g, i, \text{Mainloop} - 1))$$

**1.6: Spurious input check requiring IAF > 0****Condition I:**True**Condition II:**Params.IAltFactor ≤ 0**Condition III:**True**1.7: Spurious Input check requiring NumWEvents > 0****Condition I:**True**Condition II:**Params.NumWEvents ≤ 0**Condition III:**True

### 1.8: Negative NW value

#### Condition I:

$\exists B, g, e, t, \text{Active}(B, g) \wedge \text{Active}(\neg B, e) \wedge 1 < t < \text{Mainloop} \wedge$   
 $(\exists j, k, 1 \leq k \leq \text{Army}[\neg B, e].\text{Squadrons} \wedge 1 \leq j \leq \text{Army}[B, g].\text{Squadrons} \wedge$   
 $\text{Endurance}(B, g, j, t) > 0 \wedge \text{Endurance}(\neg B, e, k, t) > 0 \wedge \text{Observe}(B, g, j, e, k, t)) \wedge$   
 $\text{Params.NumWTypes} > 1$

#### Condition II:

$\exists i, 1 \leq i \leq \text{Params.NumWTypes}, \text{Army}[B, g].\text{WeapPriority}[e, i] < 0 \vee$   
 $\text{NumWeapon}(B, g, i, \text{Mainloop})$   
 $< (\text{NumWeapon}(B, g, i, \text{Mainloop} - 1) - \text{NumWeapon}(B, g, i, \text{Mainloop}) +$   
 $\left[ \sum_{n=1}^{\text{Mainloop} \cdot \text{NArmy}[\neg B]} \sum_{e'=1} \left( \frac{\min(|\{k' \ni \exists j, \text{Observe}(B, g, j, e', k', n-1)\}|, \text{Army}[B, g].\text{WeapPriority}[e', i]), \text{NumWeapon}(B, g, i, n))}{\text{NumWeapon}(B, g, i, \text{Mainloop} - 1)} \right) \right] \right)$

#### Condition III:

$(\exists m, 1 \leq m \leq \text{NCmsgs}[B], \text{Mimp}(B, g, m, \text{Duration}) \wedge \neg \text{Mimp}(B, g, m, \text{Mainloop} - 1)) \wedge$   
 $(\exists m, 1 \leq m \leq \text{NCmsgs}[\neg B], \text{Mimp}(\neg B, e, m, \text{Duration}) \wedge$   
 $\neg \text{Mimp}(\neg B, e, m, \text{Mainloop} - 1))$

### 1.9: PSentListLoc sends out of range squadron to SquadAlive

#### Condition I:

$\text{Active}(B, g) \wedge \text{Active}(\neg B, e) \wedge \text{Active}(B, f) \wedge \text{Army}[B, f].\text{Report} = g \wedge$   
 $(\exists t, 1 \leq t \leq \text{Duration},$   
 $t = \text{Mainloop} - \text{RepT}(B, f, g) - \text{Army}[B, g].\text{ProcDelay}$   
 $- \frac{\text{Army}[B, g].\text{RecRate}}{\text{NumRec}(B, g, \text{Mainloop} - \text{Army}[B, g].\text{ProcDelay})} \wedge$   
 $(\exists k, 1 \leq k \leq \text{Army}[\neg B, e].\text{Squadrons}, (\exists j, 1 \leq j \leq \text{Army}[B, f].\text{Squadrons},$   
 $\text{Observe}(B, f, j, e, k, t) \wedge \text{Endurance}(\neg B, e, k, t) > 0))$

#### Condition II:

$(\exists m, 1 \leq m \leq \text{NCmsgs}[\neg B], (\neg \text{Mimp}(\neg B, e, m, t)) \wedge \text{Mimp}(\neg B, e, m, \text{Mainloop}) \wedge$   
 $\text{Army}[\neg B, e].\text{Squadrons} > \text{Cmsgs}[\neg B, m].\text{msg.Squadrons}))$

#### Condition III: True



### 1.10: Restriction that SquadIntensity>0

Condition I:

$$\begin{aligned} &(\exists B, B \in \{\text{true}, \text{false}\}, N\text{Army}[B] > 0 \wedge \\ &(\exists g, g \in \{1 \dots N\text{Army}[B]\}, \text{Army}[B, g].\text{Squadrons} > 0 \\ &\wedge (\exists j, j \in \{1 \dots \text{Army}[B, g].\text{Squadrons}\}, \end{aligned}$$

condII

$$\text{Army}[B, g].\text{SquadIntensity}[j] \leq 0)))$$

Condition III:True

### 1.11: Restriction that FixSuppl $\geq 0$

Condition I:

$$\begin{aligned} &(\exists B, B \in \{\text{true}, \text{false}\}, N\text{Army}[B] > 0 \wedge \\ &(\exists g, g \in \{1 \dots N\text{Army}[B]\}, \text{Army}[B, g].\text{Squadrons} > 0 \end{aligned}$$

Condition II:

$$\text{Army}[B, g].\text{FixSuppl}))$$

Condition III:True

### 1.12: Segmentation fault when squadron leaves Terrain grid

Condition I:Active(B, g)

Condition II:

$$\begin{aligned} &(\exists j, 1 \leq j \leq \text{Army}[B, g].\text{Squadrons}, \text{Endurance}(B, g, j, \text{Mainloop}) > 0 \wedge \\ &(X_{B, g, j}(\text{Mainloop}) < 0 \vee X_{B, g, j}(\text{Mainloop}) > \text{Params.XDelta} \times \text{MaxTerrain} \vee \\ &Y_{B, g, j}(\text{Mainloop}) < 0 \vee Y_{B, g, j}(\text{Mainloop}) > \text{Params.YDelta} \times \text{MaxTerrain})) \end{aligned}$$

Condition III:True

### 1.13: Weapon use functions misordered

Condition I:

$$\begin{aligned} &\exists B, g, e, \text{Active}(B, g) \wedge \text{Active}(\neg B, e) \wedge \\ &(\exists k, 1 \leq k \leq \text{Army}[\neg B, e].\text{Squadrons}, \text{Endurance}(\neg B, e, k, \text{Mainloop}) > 0 \wedge \\ &(\exists j, 1 \leq j \leq \text{Army}[B, g].\text{Squadrons}, \text{Observe}(B, g, j, e, k, \text{Mainloop} - 1))) \end{aligned}$$

Condition II:True

Condition III:

$$\begin{aligned} &\{k' \mid (\exists j, \text{Observe}(B, g, j, e, k', \text{Mainloop}))\} \\ &\neq \{k'' \mid (\exists j, \text{Observe}(B, g, j, e, k'', \text{Mainloop} - 1))\} \wedge \\ &(\exists j, 1 \leq j \leq \text{Army}[B, g].\text{Squadrons}, \text{Endurance}(B, g, j, \text{Duration}) > 0) \wedge \\ &(\exists k, 1 \leq k \leq \text{Army}[\neg B, e].\text{Squadrons}, \text{Endurance}(\neg B, e, k, \text{Duration}) > 0) \end{aligned}$$

**1.14: Observation list reversed, causes error in firing**  
and

**1.15: Unneccesary addition of one to target list subscript in arguments to SetLLCoords**  
and

**1.16: Unneccesary adding of one to weapon subscript in arguments to SetLLCoords**  
and

**1.26: Improper targeting due to misordered observation list**  
**Condition I:**

$Active(B, g) \wedge Active(\neg B, e) \wedge$   
 $(\exists k, 1 \leq k \leq Army[\neg B, e].Squadrons, Endurance(\neg B, e, k, Duration) > 0 \wedge$   
 $(\exists j, 1 \leq j \leq Army[B, g].Squadrons, Endurance(B, g, j, Mainloop) > 0 \wedge$   
 $Observe(B, g, j, e, k, Mainloop - 1))) \wedge Params.NumWTypes > 1$

**Condition II:**

$(\exists k', 1 \leq k' \leq Army[\neg B, e].Squadrons, Endurance(\neg B, e, k', Mainloop - 1) > 0 \wedge$   
 $(\exists j, Observe(B, g, j, e, k', Mainloop - 1)) \wedge$   
 $(x_{\neg B, e, k'}(Mainloop) \neq x_{\neg B, e, k}(Mainloop) \vee y_{\neg B, e, k'}(Mainloop) \neq y_{\neg B, e, k}(Mainloop)))$

**Condition III:**

$\{k \ni (\exists j, Observe(B, g, j, e, k, Mainloop))\} \mid >$   
 $\min(Army[B, g].WeapPriority[e, 1], NumWeapon(B, g, 1, Mainloop)) \wedge$   
 $(Army[B, g].Weapon[1].Damage \neq Army[B, g].Weapon[2].Damage \vee$   
 $Army[\neg B, e].WeapSensativity[1] \neq Army[\neg B, e].WeapSensativity[2]) \wedge$   
 $(\exists m, 1 \leq m \leq NCmsgs[\neg B],$   
 $Mimp(\neg B, e, m, Duration) \wedge \neg Mimp(\neg B, e, m, Mainloop))$

**1.17: Accepts Army.Squadrons=0 as valid data**

**Condition I:**

$$(\exists B, B \in \{\text{true}, \text{false}\}, \text{NArmy}[B] > 0$$

**Condition II:**  $\text{Army}[B, g].\text{Squadrons} = 0$ )

**Condition III:** True

**1.18: TerrMoveTM returns unstable value if battalion leaves terrain grid**

**Condition I:**  $\text{Active}(B, g)$

**Condition II:**

$$(X_{B,g}(\text{Mainloop}) < 0 \vee X_{B,g}(\text{Mainloop}) > \text{Params.XDelta} \times \text{MaxTerrain} \vee \\ Y_{B,g}(\text{Mainloop}) < 0 \vee Y_{B,g}(\text{Mainloop}) > \text{Params.YDelta} \times \text{MaxTerrain})$$

**Condition III:**

$$\text{Duration} > \text{Mainloop} \wedge \\ (\nexists i, 1 \leq i \leq \text{NCmsgs}[B], \text{Mimp}(B, g, i, \text{Duration}) \wedge \neg \text{Mimp}(B, g, i, \text{Mainloop} - 1))$$

**1.19: NumCas not cleared by command message**

**Condition I:**

$$\text{Active}(B, g) \wedge \\ (\exists i, 1 \leq i \leq \text{NCmsgs}[B], \text{Mimp}(B, g, i, \text{Mainloop}) \wedge \neg \text{Mimp}(B, g, i, \text{Mainloop} - 1))$$

**Condition II:**

$$\exists j, 1 \leq j \leq \text{Army}[B, g].\text{Squadrons}, \text{Casualty}(B, g, j, \text{Mainloop})$$

**Condition III:** True

### 1.20: $NW > 0$ when $KF \leq 0$

#### Condition I:

$Active(B, g) \wedge NArmy[\neg B] > 0 \wedge Params.NumWTypes > 0 \wedge$   
 $(\exists i, 1 \leq i \leq Params.NumWTypes,$   
     $( Army[B, g].Weapon[i].NumWeapon > 0) \wedge$   
     $( Army[B, g].Weapon[i].UseLimit > 0) \wedge$   
     $( Army[B, g].Weapon[i].Range > 0) \wedge$   
 $(\exists e, 1 \leq e \leq NArmy[\neg B], Army[\neg B, e].Squadrons > 0 \wedge$   
 $(\exists k, 1 \leq k \leq Army[\neg B, e].Squadrons,$   
 $(\exists j, 1 \leq j \leq Army[B, g].Squadrons,$   
     $Endurance(\neg B, e, k, Mainloop) > 0 \wedge$   
     $Endurance(B, g, j, Mainloop) > 0 \wedge$   
     $Observe(B, g, j, e, k, Mainloop - 1) \wedge InRange(B, g, i, e, k, Mainloop)$

**Condition II:**  $(Army[B, g].Weapon[i].FireRate \leq 0)$

#### Condition III:

$(Army[B, g].Weapon[i].Damage \neq 0) \wedge (Army[\neg B, e].WeaponSensitivity[i] > 0) \wedge$   
 $(Duration > Mainloop) \wedge$   
 $(\exists r, 1 \leq r \leq NCmsgs[\neg B],$   
 $Mimp(\neg B, e, r, Duration) \wedge \neg Mimp(\neg B, e, r, Mainloop - 1)))$

### 1.22: Report Message processed ahead of command message with equal priority, receipt time

**Condition I:**

$$\begin{aligned}
 &Active(B, g) \wedge Active(\neg B, e) \wedge Active(B, f) \wedge \\
 &(\exists i, 1 \leq i \leq NCmsgs[B], Mimp(B, g, i, Mainloop) \wedge \neg Mimp(B, g, i, Mainloop - 1) \wedge \\
 &Army[B, f].Report = g \wedge (\exists t, 1 \leq t \leq Duration, \\
 &(t = Mainloop - Army[B, g].ProcDelay - Army[B, g].MediaDelay \\
 &\quad - Army[B, f].SendRate - 1) \wedge \\
 &(\exists k, 1 \leq k \leq Army[\neg B, e].Squadrons, Endurance(\neg B, e, k, t) > 0 \wedge \\
 &\quad \exists j, 1 \leq j \leq Army[B, f].Squadrons, Observe(B, f, j, e, k, t)))
 \end{aligned}$$

**Condition II:**  $Army[B, f].Priority = Cmsgs[B, i].Priority$

**Condition III:**

$$\left( \sum_{m=1}^{NCmsgs[B]} \begin{cases} 1 & \text{if } Mimp(B, g, m, Mainloop) \wedge \neg Mimp(B, g, m, Mainloop - 1) \\ 0 & \text{otherwise} \end{cases} \right) \geq NumProcess(B, g, Mainloop)$$

### 1.23: Invalid width, height when squadron leaves grid

**Condition I:**  $Active(B, g)$

**Condition II:**

$$\begin{aligned}
 &(\exists j, 1 \leq j \leq Army[B, g].Squadrons, Endurance(B, g, j, Mainloop) > 0 \wedge \\
 &(X_{B, g, j}(Mainloop) < 0 \vee X_{B, g, j}(Mainloop) > Params.XDelta \times MaxTerrain \vee \\
 &Y_{B, g, j}(Mainloop) < 0 \vee Y_{B, g, j}(Mainloop) > Params.YDelta \times MaxTerrain))
 \end{aligned}$$

**Condition III:**

$$(\exists i, 1 \leq i \leq NCmsgs[B], Mimp(B, g, i, Duration) \wedge \neg Mimp(B, g, i, Mainloop - 1))$$

### 1.25: Observations and Weapon coordinates cleared by command messages

#### Condition I:

$Active(B, g) \wedge$   
 $(\exists i, 1 \leq i \leq NCmsgs[B], Mimp(B, g, i, Mainloop) \wedge \neg Mimp(B, g, i, Mainloop - 1))$

#### Condition II:

$Active(\neg B, e) \wedge$   
 $(\exists i, 1 \leq i \leq Params.NumWTypes,$   
 $(NumWeapon(B, g, i, Mainloop) > 0 \wedge$   
 $(Army[B, g].Weapon[i].FireRate > 0) \wedge$   
 $(Army[B, g].Weapon[i].UseLimit > 0) \wedge$   
 $(Army[B, g].Weapon[i].Range > 0) \wedge$   
 $(\exists k, 1 \leq k \leq Army[\neg B, e].Squadrons,$   
 $(\exists j, 1 \leq j \leq Army[B, g].Squadrons,$   
 $Endurance(\neg B, e, k, Mainloop) > 0 \wedge$   
 $Endurance(B, g, j, Mainloop) > 0 \wedge$   
 $Observe(B, g, j, e, k, Mainloop - 1) \wedge InRange(B, g, i, e, k, Mainloop - 1))$

#### Condition III:

$(Army[B, g].Weapon[i].Damage \neq 0) \wedge (Army[\neg B, e].WeaponSensitivity[i] > 0) \wedge$   
 $(Duration > Mainloop + 1) \wedge (\neg Casualty(\neg B, e, k, Mainloop)) \wedge$   
 $0.5 > \frac{Endurance(\neg B, e, k, Mainloop) - Damage(\neg B, e, k, Mainloop) + Damage(\neg B, e, k, Mainloop - 1)}{Army[\neg B, e].Endurance[k]} \wedge$   
 $(\neg \exists r, 1 \leq r \leq NCmsgs[\neg B],$   
 $Mimp(\neg B, e, r, Duration) \wedge \neg Mimp(\neg B, e, r, Mainloop - 1))))$

### 1.27: Enemy instead of current position in observation jamming

Condition I:

$Active(\neg B, e) \wedge Active(B, g) \wedge$   
 $(\exists k, 1 \leq k \leq Army[\neg B, e].Squadrons,$   
 $(\exists j, 1 \leq j \leq Army[B, g].Squadrons,$   
 $Endurance(\neg B, e, k, Mainloop) > 0 \wedge$   
 $Endurance(B, g, j, Mainloop) > 0 \wedge BigEnough(B, g, j, e, k, t) \wedge Clear(B, g, j, e, k, t)$

Condition II:

$Params.SampleRate > 2 \wedge$   
 $(x_{\neg B, e}(Mainloop) \neq x_{B, g}(Mainloop) \vee$   
 $y_{\neg B, e}(Mainloop) \neq y_{B, g}(Mainloop)) \wedge$   
 $x_{gj} = x_{B, g, j}(t-1) \wedge y_{gj} = y_{B, g, j}(t-1) \wedge$   
 $x_{ek} = x_{\neg B, e, k}(t-1) \wedge y_{ek} = y_{\neg B, e, k}(t-1) \wedge$   
 $((\frac{BI(a', c', x_{ek}, y_{ek}) - Army[\neg B, e].SquadIntensity[k]}{BI(a', c', x_{ek}, y_{ek})} -$   
 $Params.SampleRate$   
 $\sum_{n=1} ((WO(x_{ek}, y_{ek}, Mainloop) \times Army[B, g].VWEffec$   
 $\sum_{e'=1}^{NArmy[\neg B]} \left\{ \begin{array}{ll} 0 & \text{if } \sqrt{(x_{\neg B, e'}(Mainloop-1) - x_{ek})^2 + (y_{\neg B, e'}(Mainloop-1) - y_{ek})^2} > Army[\neg B, e'].ObsJamRadius \\ \sqrt{(x_{\neg B, e'}(Mainloop-1) - x_{ek})^2 + (y_{\neg B, e'}(Mainloop-1) - y_{ek})^2} & \text{otherwise} \end{array} \right. \times Army[\neg B, e'].ObsJamEffect$   
 $)) < Army[B, g].ObsMinContrast[j])$

Condition III: True

### 1.28: Allocated fixing exceeds NumFixers×FixRate

Condition I:

$Active(B, g) \wedge (\exists j, 1 \leq j \leq Army[B, g].Squadrons, Casualty(B, g, j, Mainloop)$

Condition II:

$(Endurance(B, g, j, Mainloop) - Army[B, g].Endurance[j]) >$   
 $(Army[B, g].FixRate \times NumFix(B, g, Mainloop))$

Condition III:

$(Endurance(B, g, j, Mainloop) + Army[B, g].FixRate \times NumFix(B, g, Mainloop))$   
 $< \frac{Army[B, g].Endurance[j]}{2} \wedge$   
 $(\exists m, 1 \leq m \leq NCmsgs[B], Mimp(B, g, m, Duration) \wedge \neg Mimp(B, g, m, Mainloop))$

## **APPENDIX C**

### **FAILURE REGION-VARIABLE INCIDENCE MATRICES**

The tables in this appendix contain the results of the analysis of the failure regions of the faults from Shimeall and Leveson's study (Shimeall and Leveson, 1991). The failure regions are contained in a technical report (Shimeall, 1991).

The rows of the table are labeled with program dimensions. The columns are labeled with failure region numbers. A plus (+) after a failure region number indicates that multiple faults had exactly the same failure region; the data for the failure region analysis were entered only once in the table.

The table entries are of the form: I 5. The "I" is an artifact of the initial analysis method and is no longer of importance. The number, e.g. 5, gives the lowest numbered failure region to which that failure region is identical in its bounds for the given program dimension.



TABLE C.1(a) - VERSION 1 FAILURE REGION-VARIABLE INCIDENCE MATRIX, PART 1

Region	1.1	1.2	1.3	1.4	1.5	1.6	1.7	1.8	1.9	1.10	1.11	1.12
Army[].Endurance[]												
Army[].FixRate												
Army[].FixSuppl					15							
Army[].MediaDelay									19			
Army[].ObsJamEff												
Army[].ObsJamRadius												
Army[].ObsMinContrast[]												
Army[].Priority												
Army[].ProcDelay									19			
Army[].Report									19			
Army[].SendRate									19			
Army[].SquadIntensity[]												
Army[].Squadrons	11				15				19	110	111	115
Army[].VWEff												
Army[].WeapPriority[]								18				
Army[].WeapSensitivity[]												
Army[].Weapon[].Damage												
Army[].Weapon[].FireRate												
Army[].Weapon[].Range												
Army[].Weapon[].UseLimit												
Cmsgs[].Priority		12										
Cmsgs[].msg		12										
Cmsgs[].msg.Squadrons									19			
Duration	11	12						12	19			
NArmy[]	11							18		11	11	

TABLE C.1(b) - VERSION 1 FAILURE REGION-VARIABLE INCIDENCE MATRIX, PART 1 (continued)

Region	1.1	1.2	1.3	1.4	1.5	1.6	1.7	1.8	1.9	1.10	1.11	1.12
NCmsgs[]		112						118	119			
Params.IAltFactor						116						
Params.NumWEvents			113	113			117					
Params.NumWTypes								118				
Params.SampleRate												
Params.XDelta												1112
Params.YDelta												1112
Weather[].dWX			113									
Weather[].dWY				114								
Active		112			112			118	119			112
BI												
BigEnough												
Casualty					115							
Clear												
Damage												
Endurance								118	119			1112
InRange												
Mimp		112			115			118	119			
NumFix												
NumProcess												
NumWeapon								118				
Observe								118	118			
Repair					115							
WO												
xbg												
xbgi												1112
ybg												
ybgi												1112

TABLE C.1(c) - VERSION 1 FAILURE REGION VARIABLE INCIDENCE MATRIX, PART 2

Region	1.13	1.14+	1.17	1.18	1.19	1.20	1.22	1.23	1.25	1.27	1.28
Army [].Endurance[]									125		128
Army [].FixRate											128
Army [].FixSuppl											
Army [].MediaDelay						19					
Army [].ObsJamEff										127	
Army [].ObsJamRadius										127	
Army [].ObsMinContrast[]										127	
Army [].Priority							122				
Army [].ProcDelay							19				
Army [].Report							19				
Army [].SendRate							19			127	
Army [].SquadIntensity[]										127	
Army [].Squadrons	18	18	117		19	120	19	112	120	18	15
Army [].VWEff										127	
Army [].WeapPriority[]		114			120						
Army [].WeapSensitivity[]		114				120				120	
Army [].Weapon[].Damage		114				120				120	
Army [].Weapon[].FireRate						120				125	
Army [].Weapon[].Range						120				120	
Army [].Weapon[].UseLimit						120				120	
Cmsgs [].Priority							122				
Cmsgs [].msg											
Cmsgs [].msg.Squadrons											
Duration	113	114		118		120	19	115	125		15
NArmy []			11			120				127	

TABLE C.1(d) - VERSION 1 FAILURE REGION VARIABLE INCIDENCE MATRIX, PART 2 (continued)

Region	1.13	1.14+	1.17	1.18	1.19	1.20	1.22	1.23	1.25	1.27	1.28
NCmsgs[]		114		114	19		122	15	15		15
Params.IAltFactor											
Params.NumWEvents											
Params.NumWTypes		114				18				18	
Params.SampleRate											127
Params.XDelta				112				112			
Params.YDelta				112				112			
Weather[].dWX											
Weather[].dWY											
Active	18	18		12	12	12	19	12	18	18	12
BI										127	
BigEnough										127	
Casualty					15				125		15
Clear										127	
Damage									125		
Endurance	18	14				18	19	112	125	18	128
InRange						120			120		
Mimp		14		14	19	14	122	15	125		15
NumFix											128
NumProcess							122				
NumWeapon		14				120			120		
Observe	113	14				14	19		120		
Repair											
WO										127	
xbg				118						127	
xbgj								112		127	
ybg		14								127	
ybgj				118				112		127	
		14								127	

TABLE C.2(a) - VERSION 2 FAILURE REGION-VARIABLE INCIDENCE MATRIX, PART 1

Region	2.1	2.2	2.3	2.4	2.5	2.6+	2.7	2.8	2.10	2.11	2.12	2.13
Army[].Endurance[]												13
Army[].MediaDelay												
Army[].NumFixers												
Army[].NumJammers												
Army[].NumProcess												
Army[].NumReceive					15							
Army[].NumSend												
Army[].Priority												
Army[].ProcDelay												
Army[].Report	11											
Army[].SendRate												
Army[].SquadLength							17	18				
Army[].SquadWidth							17	18				
Army[].Squadrons	11			14	15		17	18				13
Cmsgs[].Army		12										
Cmsgs[].Dest		12		14								
Cmsgs[].Time		12										
Cmsgs[].msg.NumFixers												
Cmsgs[].msg.NumJammers												
Cmsgs[].msg.NumProcess												
Cmsgs[].msg.NumReceive					15							
Cmsgs[].msg.NumSend												
Cmsgs[].msg.Squadrons												13
Duration		12				12	17	17				15
NArmy[]	11	12			15							

TABLE C.2(b) - VERSION 2 FAILURE REGION-VARIABLE INCIDENCE MATRIX, PART 1 (continued)

Region	2.1	2.2	2.3	2.4	2.5	2.6+	2.7	2.8	2.10	2.11	2.12	2.13
NCmsgs[]		112	112	112	112							112
Params.NumWTypes					115							
Params.SampleRate											112	
Params.XDelta						116			110			
Params.YDelta						116				111		
Active	111		113	113	113	113	111	111				
Casualty				114								
Clear							117	117				
CmdSum			113									
Endurance					115		117	118				
InRange					115							
Mimp					115							113
NumProcess			113									
NumRec				114								
Observe	111				115		117	117				
Obvious												
RecT			113	113			117	117				
ReptSum			113									
SomeObserve												
xbg						116						
xbgi												
ybg						116	117	117				
ybgi							117	117				

TABLE C-2(c) - VERSION 2 FAILURE REGION-VARIABLE INCIDENCE MATRIX, PART 2

Region	2.14	2.15	2.16	2.17	2.18	2.19	2.20	2.21	2.22	2.23	2.25	2.26	2.30
Army[].Endurance[]													
Army[].MediaDelay											125		
Army[].NumFixers	114												
Army[].NumJammers		115											
Army[].NumProcess			116										
Army[].NumReceive				117									
Army[].NumSend					118								
Army[].Priority											125		
Army[].ProcDelay											125		
Army[].Report											125		
Army[].SendRate											125		
Army[].SquadLength													
Army[].SquadWidth													
Army[].Squadrons	114	115	116	117	118						125		
Cmsgs[].Army													
Cmsgs[].Dest													
Cmsgs[].Time													
Cmsgs[].msg.NumFixers						119							
Cmsgs[].msg.NumJammers							120						
Cmsgs[].msg.NumProcess								121					
Cmsgs[].msg.NumReceive									122				
Cmsgs[].msg.NumSend										123			
Cmsgs[].msg.Squadrons						119	120	121	122	123			
Duration	114	114	114	114	114	119	119	119	119	119	125		16
NArmy[]	114	114	114	114	114	119	119	119	119	119			

**TABLE C.2(d) - VERSION 2 FAILURE REGION-VARIABLE INCIDENCE MATRIX, PART 2 (continued)**

Region	2.14	2.15	2.16	2.17	2.18	2.19	2.20	2.21	2.22	2.23	2.25	2.26	2.30
NCmsgsl													
Params.NumWTypes													
Params.SampleRate													
Params.XDelta													
Params.YDelta													
Active													
Casualty													
Clear													
CmdSum													
Endurance													
InRange													
Mimp													
NumProcess													
NumRec													
Observe													
Obvious													
RecT													
ReptSum													
SomeObserve													
xbg													
xbgj													
ybg													
ybgj													



TABLE C.3(a) - VERSION 3 FAILURE REGION-VARIABLE INCIDENCE MATRIX, PART 1

Region	3.1	3.2	3.3	3.4	3.5	3.6	3.7+	3.9	3.10	3.11+	3.12	3.13	3.15	3.16	3.17
Army[].Endurance[]															
Army[].MediaDelay															
Army[].NumFixers															
Army[].NumJammers															
Army[].NumProcess	11														
Army[].NumReceive	11	11	11												
Army[].NumSend															
Army[].ProcDelay															
Army[].SquadLength												113			
Army[].SquadWidth												113			
Army[].Squadrons			13		15	16		19						19	
Army[].VO[]					15									15	
Army[].Weapon[].FireRate							17	17							
Army[].Weapon[].NumWeapon							17	17							
Army[].Weapon[].Range							17	17							
Army[].Weapon[].UseLimit							17	17							
Army[].X								19							
Army[].Y								19							
Cmsgs[].Dest															
Cmsgs[].Time															
Cmsgs[].msg.Endurance[]				14									15		
Cmsgs[].msg.NumFixers															
Cmsgs[].msg.NumJammers															
Cmsgs[].msg.NumProcess															
Cmsgs[].msg.NumReceive															
Cmsgs[].msg.NumSend															

TABLE C.3(b) - VERSION 3 FAILURE REGION-VARIABLE INCIDENCE MATRIX, PART 1 (continued)

Region	3.1	3.2	3.3	3.4	3.5	3.6	3.7+	3.9	3.10	3.11+	3.12	3.13	3.15	3.16	3.17
Cmsgs[.msg.Squadrons				14									15		
Cmsgs[.msg.V0]													15		
Duration	11		13	14	15		17	19	15	13	112		13	13	17
NArmy[]			13	14			17						13		17
NCmsgs[]	11	11	11	14	15		17	17			17		11		
Params.NumWTypes							17	17							
Params.SampleRate															
Params.XDelta										111				116	
Params.YDelta										111				116	
Active()	11	11				11	11	19		11	11	19		11	
BigEnough()															
Casualty()								19							
Clear()												13			
Damage()								19							
Endurance()			13			16	17	17					13		
InRange()							17	17							
Mimp()				14	15		17	17			17		14		
NumProcess()															
Observe()							17	17							
Obvious()												113			
Suppl()															
V()											112				
xbg()										111	112			116	
xbgj()								19				113			
ybg()										111	112			116	
ybgj()								19				113			

TABLE C.3(c) - VERSION 3 FAILURE REGION-VARIABLE INCIDENCE MATRIX, PART 2

Region	3.18+	3.21	3.22	3.23	3.24	3.25	3.26	3.28	3.32	3.33	3.34
Army[].Endurance[]							15				
Army[].MediaDelay				123							
Army[].NumFixers										133	
Army[].NumJammers											134
Army[].NumProcess											
Army[].NumReceive											
Army[].NumSend											
Army[].ProcDelay				123							
Army[].SquadLength											
Army[].SquadWidth											
Army[].Squadrons	119		119		15	13	15	15	132	133	134
Army[].V0[]						125					
Army[].Weapon[].FireRate	17										
Army[].Weapon[].NumWeapon	17										
Army[].Weapon[].Range	17										
Army[].Weapon[].UseLimit	17										
Army[].X											
Army[].Y											
Cmsgs[].Dest				123							
Cmsgs[].Time				123							
Cmsgs[].msg.Endurance[]											
Cmsgs[].msg.NumFixers											
Cmsgs[].msg.NumJammers											
Cmsgs[].msg.NumProcess											
Cmsgs[].msg.NumReceive											
Cmsgs[].msg.NumSend											

TABLE C.3(d) - VERSION 3 FAILURE REGION VARIABLE INCIDENCE MATRIX, PART2 (continued)

Region	3.18+	3.21	3.22	3.23	3.24	3.25	3.26	3.28	3.32	3.33	3.34
Cmsgs[].msg.Squadrons											
Cmsgs[].msg.V0[]											
Duration		121		15	15	15	128	121	121	121	121
NArmy[]	17	121		17	13	13	13	13	13	13	13
NCmsgs[]				11	11	14	14	14			
Params.NumWTypes	17										
Params.SampleRate											
Params.XDelta											
Params.YDelta											
Active()	11		11	11		11		11			
BigEnough()											
Casualty()			122								
Clear()											
Damage()											
Endurance()	17					13					
InRange()	17										
Mimp()				17			15	15			
NumProcess()				123							
Observe()	17										
Obvious()											
Suppl()											
V()			122								
xbg()								128			
xbgi()								128			
ybg()											
ybgi()											

TABLE C.3(e) - VERSION 3 FAILURE REGION-VARIABLE INCIDENCE MATRIX, PART 3

Region	3.35	3.36	3.37	3.38	3.39	3.40	3.41	3.42	3.43	3.44	3.45
Army[].Endurance[]											145
Army[].MediaDelay											
Army[].NumFixers											
Army[].NumJammers											
Army[].NumProcess	135										
Army[].NumReceive		136									
Army[].NumSond			137								
Army[].ProcDelay											
Army[].SquadLength											
Army[].SquadWidth											
Army[].Squadrons	135	135	135						14		145
Army[].VO[]											
Army[].Weapon[].FireRate										17	17
Army[].Weapon[].NumWeapon										17	17
Army[].Weapon[].Range										17	17
Army[].Weapon[].UseLimit										17	17
Army[].X											
Army[].Y											
Cmsgs[].Dest											
Cmsgs[].Time											
Cmsgs[].msg.Endurance[]											
Cmsgs[].msg.NumFixers											
Cmsgs[].msg.NumJammers				139							
Cmsgs[].msg.NumProcess					140						
Cmsgs[].msg.NumReceive							141				
Cmsgs[].msg.NumSend								142			

TABLE C.3(1) - VERSION 3 FAILURE REGION-VARIABLE INCIDENCE MATRIX, PART 3 (continued)

Region	3.35	3.36	3.37	3.38	3.39	3.40	3.41	3.42	3.43	3.44	3.45
Cmsgs[].msg.Squadrons				138	139	140	141	142			
Cmsgs[].msg.V0[]											
Duration	121	121	121	124	124	124	124	124			145
NArmy[]	13	13	13	17	14	14	14	14	14	17	17
NCmsgs[]				14							17
Params.NumWTypes										17	17
Params.SampleRate									143		
Params.XDelta											
Params.YDelta											
Active()									11	11	145
BigEnough()									143		
Casualty()											145
Clear()											
Damage()											145
Endurance()									17	17	145
InRange()										17	17
Mimp()				15	15	15	15	15			15
NumProcess()											
Observe()										17	17
Obvious()									143		
Suppl()											
V()											
xbgi()											
xbgi()											
ybgi()											
ybgi()											

TABLE C.4(a) - VERSION 4 FAILURE REGION-VARIABLE INCIDENCE MATRIX, PART 1

Region	4.1	4.2	4.3	4.4	4.5+	4.7	4.8	4.9	4.10	4.11	4.12
Army[].CommJamEff										111	
Army[].CommJamPriority[]										111	
Army[].CommJamRadius										111	
Army[].Endurance[]											
Army[].FixRate											
Army[].ObsJamEff											
Army[].Priority											
Army[].ProcDelay	11										
Army[].RecRate	11										
Army[].Report	11										
Army[].SendRate											
Army[].Squadrons		12	12	14	15	16	18	19	110	12	14
Army[].VWEff				14			18				
Army[].WeapPriority[]							18				
Army[].WeapSensitivity[]		13									
Army[].Weapon[].Damage		13								111	
Army[].Weapon[].Radius		13									
Army[].Weapon[].Range											
Cmsgs[].msg											
Cmsgs[].Priority											
Duration	11	12	13	14		17			110		
NArmy[]											
NCmsgs[]			13						110		
Params.NumWTypes			13				13				
Params.SampleRate				14							
Params.XDelta		12									
Params.YDelta		12									

TABLE C.4(b) - VERSION 4 FAILURE REGION-VARIABLE INCIDENCE MATRIX, PART 1 (continued)

Region	4.1	4.2	4.3	4.4	4.5+	4.7	4.8	4.9	4.10	4.11	4.12
Active	11	12	13	13	13	13	13	13	13	11	13
ax			13								
ay			13								
BigEnough				14							
Casualty									10	11	
Clear				14							
Endurance		12	12	14	15	17	18	19	10	11	14
EObserve	11			14		17	18		10		
FixRate											
InRange											
Mimp			13							11	
NumFix											
NumJam										11	
NumRec	11										
NumWeapon			13				13			11	
Observe					15	17	18	19	10	11	12
Obvious				14							
RepTs	11										
SomeObserve											
Suppl											
WO				14							
xbg										11	
xbgj		12	13	14							12
ybg										11	
ybgj		12	13	14							12



TABLE C.4(c) - VERSION 4 FAILURE REGION-VARIABLE INCIDENCE MATRIX, PART 2

Region	4.13	4.14	4.15	4.16	4.18	4.19+	4.21	4.22	4.24	4.26
Army[].CommJamEff										
Army[].CommJamPriority[]										
Army[].CommJamRadius										
Army[].Endurance[]							121			
Army[].FixRate						121				
Army[].ObsJamEff				116						
Army[].Priority		114								
Army[].ProcDelay	113	113	11							
Army[].RecRate	113	113	11							
Army[].Report	113	113	11							
Army[].SendRate	113			116			121	122		
Army[].Squadrons										
Army[].VWEff										
Army[].WeapPriority[]			18							
Army[].WeapSensitivity[]										
Army[].Weapon[].Damage										
Army[].Weapon[].Radius										
Army[].Weapon[].Range										126
Cmsgs[].msg										126
Cmsgs[].Priority										
Duration	113	113	11	116	118					
NArmy[]				116	118			122		
NCmsgs[]									124	
Params.NumWTypes			18							
Params.SampleRate								122		
Params.XDelta						119				
Params.YDelta						119				

TABLE C.4(d) - VERSION 4 FAILURE REGION-VARIABLE INCIDENCE MATRIX, PART 2 (continued)

Region	4.13	4.14	4.15	4.16	4.18	4.19+	4.21	4.22	4.24	4.26
Active	113	113	111	112	112	112	112	112	112	112
ax										
ay										
BigEnough				114				1122		
Casualty							1121			
Clear				114						
Endurance				114			1121	1122		
EObserve	113	113	115		118					
FixRate										
InRange			115							
Mimp									1124	1124
NumFix							1121			
NumJam										
NumRec	113	113	111							
NumWeapon			118							
Observe										
Obvious				116						
RepTs	113	113					1121			
SomeObserve				116						
Suppl							1121			
WO										
xbg						1119				
xbgi										
ybg						1119				
ybgi										

## APPENDIX D

### FAILURE REGION-VARIABLE INCIDENCE MATRIX ANALYSIS CODE

```
#include <stdio.h>
#define NUMVARS 70
#define NUMREG 50

main()
{
    char line[2048];
    int regid[NUMREG];
    int similar[NUMVARS][NUMREG];
    int identical[NUMVARS][NUMREG];
    int graph[NUMREG][NUMREG][NUMVARS];
    int I11[NUMREG][NUMREG];
    int S11[NUMREG][NUMREG];
    int C11[NUMREG][NUMREG];
    int N00[NUMREG][NUMREG];
    float Ijaccard[NUMREG][NUMREG];
    float Sjaccard[NUMREG][NUMREG];
    float Cjaccard[NUMREG][NUMREG];
    int I00[NUMVARS];
    int S00[NUMVARS];
    int C00[NUMVARS];
    int IjaccardCluster[NUMREG];
    int SjaccardCluster[NUMREG];
    int CjaccardCluster[NUMREG];
    float IjaccardValue[NUMREG];
    float SjaccardValue[NUMREG];
    float CjaccardValue[NUMREG];
    int temp1,temp2,temp3;
    float temp1f, temp2f, temp3f;
    int ln, col, i, j, k, f, maxline, maxreg;
    char status;

    /***** initialize arrays *****/
    for (i=0; i<=NUMVARS; i++)
    {
        I00[i] = 0;
        S00[i] = 0;
        C00[i] = 0;
        for (j=0; j<=NUMREG; j++)
```

```

        {
            identical[i][j]=0;    similar[i][j]=0;
        }
    }
for (i=0; i<=NUMREG; i++)
{
    regid[i]=0;
    IjaccardCluster[i] = 2*NUMREG;
    SjaccardCluster[i] = 2*NUMREG;
    CjaccardCluster[i] = 2*NUMREG;
    for (j=0; j<=NUMREG; j++)
    {
        I11[i][j]=0;
        S11[i][j]=0;
        C11[i][j]=0;
        N00[i][j]=0;
    }
}
ln=0; maxreg = 0;
if (fgets(line,2048,stdin)==NULL) exit(1);

/* parse the region numbers corresponding to the columns */
col=1; /* skip leading tab */
i=0;
while (col<strlen(line))
{
    i++; f=0;
    while (line[col]>='0' && line[col]<= '9')
    {
        f = f*10 + line[col] - '0';    col++;
    }
    regid[i]=f;
    if (i > maxreg) maxreg=i;
    /* skip rest of field */
    while(line[col]!='\t'&& col<strlen(line)) col++;
    col++;
}
/* now parse the body of the table */
while(!feof(stdin))
{
    ln++; i=1;                /* increment var, reset region */
    if (fgets(line, 2048, stdin)==NULL) break;
    line[strlen(line)-1]='\0';    /* clear \n from line */
    col=0;
    while (col < strlen(line)&& line[col]!='\t')
        col++;    /* skip line label */
    col++;                /* move to start of first field */
    while (col < strlen(line))
    {
        while (line[col]=='\t') /* skip over empty fields */
        {

```

```

        col++; i++;
    }
    if (line[col]!='\0')    /* if at end of line, get out of loop */
    {
        status=line[col];
        col++;
        if (line[col]!='\t')
        {
            f=0;
            /* parse for region number after I or S */
            while (line[col]>='0' && line[col]<='9')
            {
                f = f * 10 + line[col] - '0'; col++;
            }
            /* store entry in appropriate table */
            if (status == 'I')
            {
                if(identical[ln][f] < f && identical[ln][f] > 0)
                    identical[ln][regid[i]] = identical[ln][f];
                else identical[ln][regid[i]]=f;
                if (similar[ln][f]!=0)
                    similar[ln][regid[i]]=similar[ln][f];
            }
            else if (status == 'S')
            {
                if (identical[ln][f]<f && identical[ln][f]>0)
                    similar[ln][regid[i]]=identical[ln][f];
                else similar[ln][regid[i]]=f;
            }
            else if (status == 'U') /* treat as isolated identical */
            { identical[ln][regid[i]] = regid[i];
            }
            } else if (status == 'U') /* treat as isolated identical */
            { identical[ln][regid[i]] = regid[i];
            while (line[col]!='\t' && col<strlen(line))
                col++;
            /* skip rest of entry */
        }
    }
    maxline = ln;
}
/* determine if occurrences are identical, similar, or coincidental */

for (i=2; i<= maxreg; i++ )    /* compare regions pairwise */
    for (j=1; j<=(i-1); j++)
    {
        for (k=1; k<= maxline; k++) /* for each pair of regions,
                                     consider each variable */
        {
            /* if the variable occurrences are identical, graph = 3) */
            if( ( identical[k][regid[j]] != 0)
                && ( identical[k][regid[j]] == identical[k][regid[i]] ) )
                graph[regid[i]][regid[j]][k] = 3;
        }
    }

```

```

else /* if the variable occurrences are similar, graph = 2) */
    if( ( similar[k][regid[i]] != 0)
        && (similar[k][regid[i]] == regid[j] ) )
        graph[regid[i]][regid[j]][k] = 2;
    else /* if the variable occurrences are coincidental,
                                                graph = 1) */
        if( ( identical[k][regid[j]] != 0
            || similar[k][regid[j]] != 0 )
            && ( identical[k][regid[i]] != 0
            || similar[k][regid[i]] != 0 ) )
            graph[regid[i]][regid[j]][k] = 1;
        else /* the variables are not coincident in this
            pair of regions, graph = 0 */
            graph[regid[i]][regid[j]][k] = 0;
    }
}
/* compute proximity indices and coefficients */

for (i=2; i<= maxreg; i++) /* compare regions pairwise */
    for (j=1; j<=(i-1); j++)
    {
        for (k=1; k<= maxline; k++) /* for each pair of regions,
            consider each variable */
        {
            if( graph[regid[i]][regid[j]][k] == 3 ) I11[regid[i]][regid[j]]++; /
            * variables in regions i,j that are identical */
            if( graph[regid[i]][regid[j]][k] == 1 ) C11[regid[i]][regid[j]]++; /
            * variables in regions i,j that are coincidental*/
            if( identical[k][regid[j]] == 0
                && similar[k][regid[j]] == 0 /* variables that appear in
                && identical[k][regid[i]] == 0 /* neither region i nor j */
                && similar[k][regid[i]] == 0 )
                N00[regid[i]][regid[j]]++;
        }
    }
Ijaccard[regid[i]][regid[j]] = (1.0 * (I11[regid[i]][regid[j]])) /
    (1.0 * ( maxline - N00[regid[i]][regid[j]] ) );

Cjaccard[regid[i]][regid[j]] = (1.0 * (C11[regid[i]][regid[j]]))
    / (1.0 * ( maxline - N00[regid[i]][regid[j]] ) );
}

/*****order Jaccard coefficients *****/

temp1f = 1.0;
temp2f = 0.9999;

/*****Identical coefficients*****/
while(temp1f >= 0.0)
{
    for (i=2; i<= 21; i++) /* compare regions pairwise */
        for (j=1; j<=(i-1); j++)
            if( (Ijaccard[regid[i]][regid[j]] <= temp1f) &&

```

```

(Ijaccard[regid[i]][regid[j]] > temp2f) )
{
    if(I11[regid[i]][regid[j]] == 0)
        I00[(maxline-N00[regid[i]][regid[j]])]++;
    else
        printf("Ijaccard[%d][%d] = %f , Fraction = %d / %d \n",
            regid[i],regid[j],Ijaccard[regid[i]][regid[j]],
            I11[regid[i]][regid[j]],(maxline-N00[regid[i]][regid[j]]));
        f=NUMREG + 1;
        /* determine the order in which the */
        for(k=1;k<=NUMREG;k++) /* regions appear */
            if(IjaccardCluster[k] == regid[i]) f=k;
        if(f > NUMREG)
            for(k=1;k<=NUMREG;k++)
            {
                if(IjaccardCluster[k] > NUMREG)
                {
                    IjaccardCluster[k] = regid[i];
                    IjaccardValue[k] = Ijaccard[regid[i]][regid[j]];
                }
                if(IjaccardCluster[k] == regid[i]) break;
            }
        f=NUMREG + 1;
        for(k=1;k<=NUMREG;k++)
            if(IjaccardCluster[k] == regid[j]) f=k;
        if(f > NUMREG)
            for(k=1;k<=NUMREG;k++)
            {
                if(IjaccardCluster[k] > NUMREG)
                {
                    IjaccardCluster[k] = regid[j];
                    IjaccardValue[k] = Ijaccard[regid[i]][regid[j]];
                }
                if(IjaccardCluster[k] == regid[j]) break;
            }
    }
    temp1f = temp2f;
    temp2f -= 0.0001;
}

for(i=1;i<=maxline;i++) printf("I00[%d] = %d \n",i,I00[i]);
for(i=1;i<=21;i++)
    printf("IjaccardCluster[%d] = %d , IjaccardValue[%d] = %f \n",
        i,IjaccardCluster[i],i,IjaccardValue[i]);

temp1=21;
temp1f = 1.0;
temp2f = 0.9999;

/******Coincidental coefficients******/
while(temp1f >= 0.0)
{
    for (i=2; i<=temp1; i++ ) /* compare regions pairwise */

```

```

for (j=1; j<=(i-1); j++)
    if( (Cjaccard[regid[i]][regid[j]] <= temp1f) &&
        (Cjaccard[regid[i]][regid[j]] > temp2f) )
    {
        printf("Cjaccard[%d][%d] = %f , Fraction = %d / %d \n",
            regid[i],regid[j],Cjaccard[regid[i]][regid[j]],
            C11[regid[i]][regid[j]],(maxline-N00[regid[i]][regid[j]]));

        f=NUMREG + 1; /* determine the order in which the */
        for(k=1;k<=NUMREG;k++) /* regions appear */
            if(CjaccardCluster[k] == regid[i]) f=k;
        if(f > NUMREG)
            for(k=1;k<=NUMREG;k++)
            {
                if(CjaccardCluster[k] > NUMREG)
                {
                    CjaccardCluster[k] = regid[i];
                    CjaccardValue[k] = Cjaccard[regid[i]][regid[j]];
                }
                if(CjaccardCluster[k] == regid[i]) break;
            }
        f=NUMREG + 1;
        for(k=1;k<=NUMREG;k++)
            if(CjaccardCluster[k] == regid[j]) f=k;
        if(f > NUMREG)
            for(k=1;k<=NUMREG;k++)
            {
                if(CjaccardCluster[k] > NUMREG)
                {
                    CjaccardCluster[k] = regid[j];
                    CjaccardValue[k] = Cjaccard[regid[i]][regid[j]];
                }
                if(CjaccardCluster[k] == regid[j]) break;
            }
    }
    temp1f = temp2f;
    temp2f -= 0.0001;
}

for(i=1;i<=temp1;i++)
    printf("CjaccardCluster[%d] = %d , CjaccardValue[%d] = %f \n",
        i,CjaccardCluster[i],i,CjaccardValue[i]);

exit(0);
}

```



## APPENDIX E

### THRESHOLD GRAPH EDGE LISTINGS

This appendix lists the weights of the edges in the graphs for the four experimental versions. Only nonzero edges are listed. The weights are presented in two forms: a decimal fraction and a ratio of two integers. The ratio represents the exact weight; the decimal fractions were used for ordering the magnitudes of the edges.

$I [ ] [ ]$  is the coefficient for the Identical dimension.

$C [ ] [ ]$  is the coefficient for the Coincidental dimension.

$J [ ] [ ]$  is the coefficient for the Composite dimension.

Also presented is the order in which the nodes were connected in the graph and the threshold values at which they were connected, i.e., the value of their largest weighted incident edge.

$\text{Node}[ ]$  gives the number of the failure region being connected in the graph.

$I [ ]$ ,  $C [ ]$ , and  $J [ ]$  give the threshold value at which the first edge is added to that node.

**VERSION 1  
IDENTICAL**

I[23][12] = 0.700000 ; 7 / 10  
 I[11][10] = 0.500000 ; 1 / 2  
 I[17][10] = 0.500000 ; 1 / 2  
 I[17][11] = 0.500000 ; 1 / 2  
 I[22][9] = 0.466667 ; 7 / 15  
 I[4][3] = 0.333333 ; 1 / 3  
 I[10][1] = 0.333333 ; 1 / 3  
 I[11][1] = 0.333333 ; 1 / 3  
 I[17][1] = 0.333333 ; 1 / 3  
 I[28][23] = 0.333333 ; 5 / 15  
 I[28][5] = 0.307692 ; 4 / 13  
 I[19][5] = 0.285714 ; 2 / 7  
 I[18][12] = 0.250000 ; 3 / 12  
 I[23][18] = 0.250000 ; 3 / 12  
 I[23][5] = 0.230769 ; 3 / 13  
 I[27][20] = 0.214286 ; 6 / 28  
 I[12][5] = 0.181818 ; 2 / 11  
 I[13][8] = 0.181818 ; 2 / 11  
 I[28][19] = 0.181818 ; 2 / 11  
 I[14][13] = 0.142857 ; 2 / 14  
 I[28][12] = 0.133333 ; 2 / 15  
 I[19][2] = 0.125000 ; 1 / 8  
 I[27][13] = 0.120000 ; 3 / 25  
 I[20][8] = 0.117647 ; 2 / 17  
 I[25][20] = 0.105263 ; 2 / 19  
 I[27][8] = 0.103448 ; 3 / 29  
 I[5][2] = 0.100000 ; 1 / 10  
 I[18][2] = 0.100000 ; 1 / 10  
 I[19][12] = 0.100000 ; 1 / 10  
 I[19][18] = 0.100000 ; 1 / 10  
 I[20][18] = 0.100000 ; 2 / 20  
 I[23][19] = 0.090909 ; 1 / 11  
 I[8][2] = 0.083333 ; 1 / 12  
 I[12][2] = 0.083333 ; 1 / 12  
 I[18][5] = 0.083333 ; 1 / 12  
 I[23][2] = 0.083333 ; 1 / 12  
 I[25][13] = 0.083333 ; 1 / 12  
 I[19][9] = 0.076923 ; 1 / 13  
 I[28][2] = 0.076923 ; 1 / 13  
 I[27][14] = 0.068966 ; 2 / 29  
 I[14][8] = 0.066667 ; 1 / 15  
 I[20][13] = 0.066667 ; 1 / 15  
 I[22][19] = 0.066667 ; 1 / 15  
 I[25][8] = 0.066667 ; 1 / 15  
 I[28][18] = 0.066667 ; 1 / 15  
 I[28][25] = 0.066667 ; 1 / 15  
 I[9][8] = 0.062500 ; 1 / 16

I[25][23] = 0.062500 ; 1 / 16  
 I[20][19] = 0.058824 ; 1 / 17  
 I[20][2] = 0.055556 ; 1 / 18  
 I[20][5] = 0.055556 ; 1 / 18  
 I[22][8] = 0.055556 ; 1 / 18  
 I[25][14] = 0.055556 ; 1 / 18  
 I[20][12] = 0.052632 ; 1 / 19  
 I[23][20] = 0.050000 ; 1 / 20  
 I[28][20] = 0.047619 ; 1 / 21  
 I[27][25] = 0.031250 ; 1 / 32  
 ALL OTHER EDGES = 0.000000

Node[1] = 23 , I[1] = 0.700000  
 Node[2] = 12 , I[2] = 0.700000  
 Node[3] = 11 , I[3] = 0.500000  
 Node[4] = 10 , I[4] = 0.500000  
 Node[5] = 17 , I[5] = 0.500000  
 Node[6] = 22 , I[6] = 0.466667  
 Node[7] = 9 , I[7] = 0.466667  
 Node[8] = 4 , I[8] = 0.333333  
 Node[9] = 3 , I[9] = 0.333333  
 Node[10] = 1 , I[10] = 0.333333  
 Node[11] = 28 , I[11] = 0.333333  
 Node[12] = 5 , I[12] = 0.307692  
 Node[13] = 19 , I[13] = 0.285714  
 Node[14] = 18 , I[14] = 0.250000  
 Node[15] = 27 , I[15] = 0.214286  
 Node[16] = 20 , I[16] = 0.214286  
 Node[17] = 13 , I[17] = 0.181818  
 Node[18] = 8 , I[18] = 0.181818  
 Node[19] = 14 , I[19] = 0.142857  
 Node[20] = 2 , I[20] = 0.125000  
 Node[21] = 25 , I[21] = 0.105263  
 Node[22] = 6 , I[22] = 0.000000  
 Node[23] = 7 , I[23] = 0.000000

**VERSION 1  
COINCIDENTAL**

C[14][8] = 0.533333 ; 8 / 15  
 C[20][14] = 0.526316 ; 10 / 19  
 C[7][3] = 0.500000 ; 1 / 2  
 C[7][4] = 0.500000 ; 1 / 2  
 C[11][10] = 0.500000 ; 1 / 2  
 C[17][10] = 0.500000 ; 1 / 2  
 C[17][11] = 0.500000 ; 1 / 2  
 C[23][14] = 0.500000 ; 8 / 16  
 C[28][25] = 0.466667 ; 7 / 15  
 C[13][9] = 0.416667 ; 5 / 12  
 C[25][19] = 0.416667 ; 5 / 12  
 C[25][9] = 0.411765 ; 7 / 17  
 C[25][8] = 0.400000 ; 6 / 15  
 C[25][14] = 0.388889 ; 7 / 18  
 C[23][9] = 0.375000 ; 6 / 16  
 C[14][9] = 0.368421 ; 7 / 19  
 C[25][22] = 0.368421 ; 7 / 19  
 C[23][13] = 0.363636 ; 4 / 11  
 C[22][13] = 0.357143 ; 5 / 14  
 C[20][8] = 0.352941 ; 6 / 17  
 C[28][9] = 0.352941 ; 6 / 17  
 C[10][1] = 0.333333 ; 1 / 3  
 C[11][1] = 0.333333 ; 1 / 3  
 C[13][1] = 0.333333 ; 2 / 6  
 C[13][12] = 0.333333 ; 3 / 9  
 C[17][1] = 0.333333 ; 1 / 3  
 C[22][2] = 0.333333 ; 5 / 15  
 C[22][14] = 0.333333 ; 7 / 21  
 C[23][8] = 0.333333 ; 5 / 15  
 C[23][22] = 0.333333 ; 6 / 18  
 C[25][13] = 0.333333 ; 4 / 12  
 C[28][13] = 0.333333 ; 4 / 12  
 C[25][20] = 0.315789 ; 6 / 19  
 C[28][14] = 0.315789 ; 6 / 19  
 C[28][22] = 0.315789 ; 6 / 19  
 C[9][8] = 0.312500 ; 5 / 16  
 C[14][12] = 0.312500 ; 5 / 16  
 C[25][23] = 0.312500 ; 5 / 16  
 C[28][9] = 0.312500 ; 5 / 16  
 C[18][2] = 0.300000 ; 3 / 10  
 C[9][2] = 0.285714 ; 4 / 14  
 C[18][8] = 0.285714 ; 4 / 14  
 C[19][5] = 0.285714 ; 2 / 7  
 C[20][9] = 0.285714 ; 6 / 21  
 C[25][2] = 0.285714 ; 4 / 14  
 C[25][5] = 0.285714 ; 4 / 14  
 C[22][8] = 0.277778 ; 5 / 18

C[23][19] = 0.272727 ; 3 / 11  
 C[28][19] = 0.272727 ; 3 / 11  
 C[19][14] = 0.266667 ; 4 / 15  
 C[20][13] = 0.266667 ; 4 / 15  
 C[22][9] = 0.266667 ; 4 / 15  
 C[22][20] = 0.260870 ; 6 / 23  
 C[8][2] = 0.250000 ; 3 / 12  
 C[14][2] = 0.250000 ; 4 / 16  
 C[18][9] = 0.250000 ; 4 / 16  
 C[19][2] = 0.250000 ; 2 / 8  
 C[19][8] = 0.250000 ; 3 / 12  
 C[19][13] = 0.250000 ; 2 / 8  
 C[23][2] = 0.250000 ; 3 / 12  
 C[23][18] = 0.250000 ; 3 / 12  
 C[25][18] = 0.250000 ; 4 / 16  
 C[19][9] = 0.230769 ; 3 / 13  
 C[28][2] = 0.230769 ; 3 / 13  
 C[13][2] = 0.222222 ; 2 / 9  
 C[13][5] = 0.222222 ; 2 / 9  
 C[18][14] = 0.222222 ; 4 / 18  
 C[22][18] = 0.222222 ; 4 / 18  
 C[14][13] = 0.214286 ; 3 / 14  
 C[27][14] = 0.206897 ; 6 / 29  
 C[9][5] = 0.200000 ; 3 / 15  
 C[19][18] = 0.200000 ; 2 / 10  
 C[20][1] = 0.200000 ; 3 / 15  
 C[22][19] = 0.200000 ; 3 / 15  
 C[23][20] = 0.200000 ; 4 / 20  
 C[27][12] = 0.200000 ; 5 / 25  
 C[28][18] = 0.200000 ; 3 / 15  
 C[28][20] = 0.190476 ; 4 / 21  
 C[12][9] = 0.187500 ; 3 / 16  
 C[25][12] = 0.187500 ; 3 / 16  
 C[8][1] = 0.181818 ; 2 / 11  
 C[13][8] = 0.181818 ; 2 / 11  
 C[18][13] = 0.181818 ; 2 / 11  
 C[23][1] = 0.181818 ; 2 / 11  
 C[27][23] = 0.178571 ; 5 / 28  
 C[14][5] = 0.176471 ; 3 / 17  
 C[22][5] = 0.176471 ; 3 / 17  
 C[13][10] = 0.166667 ; 1 / 6  
 C[13][11] = 0.166667 ; 1 / 6  
 C[17][13] = 0.166667 ; 1 / 6  
 C[19][10] = 0.166667 ; 1 / 6  
 C[19][11] = 0.166667 ; 1 / 6  
 C[19][17] = 0.166667 ; 1 / 6  
 C[22][12] = 0.166667 ; 3 / 18  
 C[28][1] = 0.166667 ; 2 / 12  
 C[9][1] = 0.153846 ; 2 / 13  
 C[25][1] = 0.153846 ; 2 / 13  
 C[8][5] = 0.142857 ; 2 / 14

C[10][5] = 0.142857 ; 1 / 7  
 C[11][5] = 0.142857 ; 1 / 7  
 C[17][5] = 0.142857 ; 1 / 7  
 C[19][1] = 0.142857 ; 1 / 7  
 C[27][20] = 0.142857 ; 4 / 28  
 C[12][8] = 0.133333 ; 2 / 15  
 C[14][1] = 0.133333 ; 2 / 15  
 C[20][10] = 0.133333 ; 2 / 15  
 C[20][11] = 0.133333 ; 2 / 15  
 C[20][17] = 0.133333 ; 2 / 15  
 C[22][1] = 0.133333 ; 2 / 15  
 C[2][1] = 0.125000 ; 1 / 8  
 C[5][1] = 0.125000 ; 1 / 8  
 C[12][10] = 0.125000 ; 1 / 8  
 C[12][11] = 0.125000 ; 1 / 8  
 C[17][12] = 0.125000 ; 1 / 8  
 C[20][19] = 0.117647 ; 2 / 17  
 C[12][1] = 0.111111 ; 1 / 9  
 C[20][2] = 0.111111 ; 2 / 18  
 C[20][5] = 0.111111 ; 2 / 18  
 C[27][18] = 0.107143 ; 3 / 28  
 C[20][12] = 0.105263 ; 2 / 19  
 C[5][2] = 0.100000 ; 1 / 10  
 C[18][1] = 0.100000 ; 1 / 10  
 C[19][12] = 0.100000 ; 1 / 10  
 C[28][27] = 0.096774 ; 3 / 31  
 C[27][9] = 0.093750 ; 3 / 32  
 C[10][8] = 0.090909 ; 1 / 11  
 C[11][8] = 0.090909 ; 1 / 11  
 C[17][8] = 0.090909 ; 1 / 11  
 C[23][10] = 0.090909 ; 1 / 11  
 C[23][11] = 0.090909 ; 1 / 11  
 C[23][17] = 0.090909 ; 1 / 11  
 C[27][22] = 0.088235 ; 3 / 34  
 C[27][10] = 0.086957 ; 2 / 23  
 C[27][11] = 0.086957 ; 2 / 23  
 C[27][17] = 0.086957 ; 2 / 23  
 C[18][5] = 0.083333 ; 1 / 12  
 C[27][1] = 0.083333 ; 2 / 24  
 C[28][10] = 0.083333 ; 1 / 12  
 C[28][11] = 0.083333 ; 1 / 12  
 C[28][17] = 0.083333 ; 1 / 12  
 C[10][9] = 0.076923 ; 1 / 13  
 C[11][9] = 0.076923 ; 1 / 13  
 C[17][9] = 0.076923 ; 1 / 13  
 C[25][10] = 0.076923 ; 1 / 13  
 C[25][11] = 0.076923 ; 1 / 13  
 C[25][17] = 0.076923 ; 1 / 13  
 C[27][19] = 0.076923 ; 2 / 26  
 C[27][5] = 0.074074 ; 2 / 27  
 C[14][10] = 0.066667 ; 1 / 15

C[14][11] = 0.066667 ; 1 / 15  
 C[17][14] = 0.066667 ; 1 / 15  
 C[22][10] = 0.066667 ; 1 / 15  
 C[22][11] = 0.066667 ; 1 / 15  
 C[22][17] = 0.066667 ; 1 / 15  
 C[28][12] = 0.066667 ; 1 / 15  
 C[28][23] = 0.066667 ; 1 / 15  
 C[27][25] = 0.062500 ; 2 / 32  
 C[20][18] = 0.050000 ; 1 / 20  
 C[27][2] = 0.035714 ; 1 / 28  
 C[27][8] = 0.034483 ; 1 / 29  
 ALL OTHER EDGES = 0.000000

Node[1] = 14 , C[1] = 0.533333  
 Node[2] = 8 , C[2] = 0.533333  
 Node[3] = 20 , C[3] = 0.526316  
 Node[4] = 7 , C[4] = 0.500000  
 Node[5] = 3 , C[5] = 0.500000  
 Node[6] = 4 , C[6] = 0.500000  
 Node[7] = 11 , C[7] = 0.500000  
 Node[8] = 10 , C[8] = 0.500000  
 Node[9] = 17 , C[9] = 0.500000  
 Node[10] = 23 , C[10] = 0.500000  
 Node[11] = 28 , C[11] = 0.466667  
 Node[12] = 25 , C[12] = 0.466667  
 Node[13] = 13 , C[13] = 0.416667  
 Node[14] = 9 , C[14] = 0.416667  
 Node[15] = 19 , C[15] = 0.416667  
 Node[16] = 22 , C[16] = 0.368421  
 Node[17] = 1 , C[17] = 0.333333  
 Node[18] = 12 , C[18] = 0.333333  
 Node[19] = 2 , C[19] = 0.333333  
 Node[20] = 18 , C[20] = 0.300000  
 Node[21] = 5 , C[21] = 0.285714  
 Node[22] = 27 , C[22] = 0.206897  
 Node[23] = 6 , C[23] = 0.000000

**VERSION 1  
COMPOSITE**

J[11][10] = 1.000000 ; 2 / 2  
 J[17][10] = 1.000000 ; 2 / 2  
 J[17][11] = 1.000000 ; 2 / 2  
 J[22][9] = 0.733333 ; 11 / 15  
 J[23][12] = 0.700000 ; 7 / 10  
 J[10][1] = 0.666667 ; 2 / 3  
 J[11][1] = 0.666667 ; 2 / 3  
 J[17][1] = 0.666667 ; 2 / 3  
 J[14][8] = 0.600000 ; 9 / 15  
 J[19][5] = 0.571429 ; 4 / 7  
 J[25][8] = 0.533333 ; 8 / 15  
 J[28][25] = 0.533333 ; 8 / 15  
 J[20][8] = 0.529412 ; 9 / 17  
 J[20][14] = 0.526316 ; 10 / 19  
 J[7][3] = 0.500000 ; 1 / 2  
 J[7][4] = 0.500000 ; 1 / 2  
 J[23][14] = 0.500000 ; 8 / 16  
 J[23][18] = 0.500000 ; 6 / 12  
 J[28][19] = 0.454545 ; 5 / 11  
 J[25][14] = 0.444444 ; 8 / 18  
 J[25][20] = 0.421053 ; 8 / 19  
 J[13][9] = 0.416667 ; 5 / 12  
 J[25][13] = 0.416667 ; 5 / 12  
 J[25][19] = 0.416667 ; 5 / 12  
 J[25][9] = 0.411765 ; 7 / 17  
 J[18][2] = 0.400000 ; 4 / 10  
 J[28][23] = 0.400000 ; 6 / 15  
 J[9][8] = 0.375000 ; 6 / 16  
 J[19][2] = 0.375000 ; 3 / 8  
 J[23][9] = 0.375000 ; 6 / 16  
 J[25][23] = 0.375000 ; 6 / 16  
 J[14][9] = 0.368421 ; 7 / 19  
 J[25][22] = 0.368421 ; 7 / 19  
 J[13][8] = 0.363636 ; 4 / 11  
 J[23][13] = 0.363636 ; 4 / 11  
 J[23][19] = 0.363636 ; 4 / 11  
 J[14][13] = 0.357143 ; 5 / 14  
 J[22][13] = 0.357143 ; 5 / 14  
 J[27][20] = 0.357143 ; 10 / 28  
 J[28][9] = 0.352941 ; 6 / 17  
 J[4][3] = 0.333333 ; 1 / 3  
 J[8][2] = 0.333333 ; 4 / 12  
 J[13][1] = 0.333333 ; 2 / 6  
 J[13][12] = 0.333333 ; 3 / 9  
 J[20][13] = 0.333333 ; 5 / 15  
 J[22][2] = 0.333333 ; 5 / 15

J[22][8] = 0.333333 ; 6 / 18  
 J[22][14] = 0.333333 ; 7 / 21  
 J[23][2] = 0.333333 ; 4 / 12  
 J[23][8] = 0.333333 ; 5 / 15  
 J[23][22] = 0.333333 ; 6 / 18  
 J[28][13] = 0.333333 ; 4 / 12  
 J[28][14] = 0.315789 ; 6 / 19  
 J[28][22] = 0.315789 ; 6 / 19  
 J[14][12] = 0.312500 ; 5 / 16  
 J[28][8] = 0.312500 ; 5 / 16  
 J[19][9] = 0.307692 ; 4 / 13  
 J[28][2] = 0.307692 ; 4 / 13  
 J[28][5] = 0.307692 ; 4 / 13  
 J[19][18] = 0.300000 ; 3 / 10  
 J[9][2] = 0.285714 ; 4 / 14  
 J[18][8] = 0.285714 ; 4 / 14  
 J[20][9] = 0.285714 ; 6 / 21  
 J[25][2] = 0.285714 ; 4 / 14  
 J[25][5] = 0.285714 ; 4 / 14  
 J[27][14] = 0.275862 ; 8 / 29  
 J[19][14] = 0.266667 ; 4 / 15  
 J[22][19] = 0.266667 ; 4 / 15  
 J[28][18] = 0.266667 ; 4 / 15  
 J[22][20] = 0.260870 ; 6 / 23  
 J[14][2] = 0.250000 ; 4 / 16  
 J[18][5] = 0.250000 ; 3 / 12  
 J[18][9] = 0.250000 ; 4 / 16  
 J[18][12] = 0.250000 ; 3 / 12  
 J[19][8] = 0.250000 ; 3 / 12  
 J[19][13] = 0.250000 ; 2 / 8  
 J[23][20] = 0.250000 ; 5 / 20  
 J[25][18] = 0.250000 ; 4 / 16  
 J[28][20] = 0.238095 ; 5 / 21  
 J[14][5] = 0.235294 ; 4 / 17  
 J[23][5] = 0.230769 ; 3 / 13  
 J[13][2] = 0.222222 ; 2 / 9  
 J[13][5] = 0.222222 ; 2 / 9  
 J[18][14] = 0.222222 ; 4 / 18  
 J[22][18] = 0.222222 ; 4 / 18  
 J[5][2] = 0.200000 ; 2 / 10  
 J[9][5] = 0.200000 ; 3 / 15  
 J[19][12] = 0.200000 ; 2 / 10  
 J[20][1] = 0.200000 ; 3 / 15  
 J[27][12] = 0.200000 ; 5 / 25  
 J[28][12] = 0.200000 ; 3 / 15  
 J[12][9] = 0.187500 ; 3 / 16  
 J[25][12] = 0.187500 ; 3 / 16  
 J[8][1] = 0.181818 ; 2 / 11  
 J[12][5] = 0.181818 ; 2 / 11  
 J[18][13] = 0.181818 ; 2 / 11  
 J[23][1] = 0.181818 ; 2 / 11

J[27][23] = 0.178571 ; 5 / 28  
 J[20][19] = 0.176471 ; 3 / 17  
 J[22][5] = 0.176471 ; 3 / 17  
 J[13][10] = 0.166667 ; 1 / 6  
 J[13][11] = 0.166667 ; 1 / 6  
 J[17][13] = 0.166667 ; 1 / 6  
 J[19][10] = 0.166667 ; 1 / 6  
 J[19][11] = 0.166667 ; 1 / 6  
 J[19][17] = 0.166667 ; 1 / 6  
 J[20][2] = 0.166667 ; 3 / 18  
 J[20][5] = 0.166667 ; 3 / 18  
 J[22][12] = 0.166667 ; 3 / 18  
 J[28][1] = 0.166667 ; 2 / 12  
 J[20][12] = 0.157895 ; 3 / 19  
 J[9][1] = 0.153846 ; 2 / 13  
 J[25][1] = 0.153846 ; 2 / 13  
 J[20][18] = 0.150000 ; 3 / 20  
 J[8][5] = 0.142857 ; 2 / 14  
 J[10][5] = 0.142857 ; 1 / 7  
 J[11][5] = 0.142857 ; 1 / 7  
 J[17][5] = 0.142857 ; 1 / 7  
 J[19][1] = 0.142857 ; 1 / 7  
 J[27][8] = 0.137931 ; 4 / 29  
 J[12][8] = 0.133333 ; 2 / 15  
 J[14][1] = 0.133333 ; 2 / 15  
 J[20][10] = 0.133333 ; 2 / 15  
 J[20][11] = 0.133333 ; 2 / 15  
 J[20][17] = 0.133333 ; 2 / 15  
 J[22][1] = 0.133333 ; 2 / 15  
 J[2][1] = 0.125000 ; 1 / 8  
 J[5][1] = 0.125000 ; 1 / 8  
 J[12][10] = 0.125000 ; 1 / 8  
 J[12][11] = 0.125000 ; 1 / 8  
 J[17][12] = 0.125000 ; 1 / 8  
 J[27][13] = 0.120000 ; 3 / 25  
 J[12][1] = 0.111111 ; 1 / 9  
 J[27][18] = 0.107143 ; 3 / 28  
 J[18][1] = 0.100000 ; 1 / 10  
 J[28][27] = 0.096774 ; 3 / 31  
 J[27][9] = 0.093750 ; 3 / 32  
 J[27][25] = 0.093750 ; 3 / 32  
 J[10][8] = 0.090909 ; 1 / 11  
 J[11][8] = 0.090909 ; 1 / 11  
 J[17][8] = 0.090909 ; 1 / 11  
 J[23][10] = 0.090909 ; 1 / 11  
 J[23][11] = 0.090909 ; 1 / 11  
 J[23][17] = 0.090909 ; 1 / 11  
 J[27][22] = 0.088235 ; 3 / 34  
 J[27][10] = 0.086957 ; 2 / 23  
 J[27][11] = 0.086957 ; 2 / 23  
 J[27][17] = 0.086957 ; 2 / 23

J[12][2] = 0.083333 ; 1 / 12  
 J[27][1] = 0.083333 ; 2 / 24  
 J[28][10] = 0.083333 ; 1 / 12  
 J[28][11] = 0.083333 ; 1 / 12  
 J[28][17] = 0.083333 ; 1 / 12  
 J[10][9] = 0.076923 ; 1 / 13  
 J[11][9] = 0.076923 ; 1 / 13  
 J[17][9] = 0.076923 ; 1 / 13  
 J[25][10] = 0.076923 ; 1 / 13  
 J[25][11] = 0.076923 ; 1 / 13  
 J[25][17] = 0.076923 ; 1 / 13  
 J[27][19] = 0.076923 ; 2 / 26  
 J[27][5] = 0.074074 ; 2 / 27  
 J[14][10] = 0.066667 ; 1 / 15  
 J[14][11] = 0.066667 ; 1 / 15  
 J[17][14] = 0.066667 ; 1 / 15  
 J[22][10] = 0.066667 ; 1 / 15  
 J[22][11] = 0.066667 ; 1 / 15  
 J[22][17] = 0.066667 ; 1 / 15  
 J[27][2] = 0.035714 ; 1 / 28  
 ALL OTHER EDGES = 0.000000

Node[1] = 11 , J[1] = 1.000000  
 Node[2] = 10 , J[2] = 1.000000  
 Node[3] = 17 , J[3] = 1.000000  
 Node[4] = 22 , J[4] = 0.733333  
 Node[5] = 9 , J[5] = 0.733333  
 Node[6] = 23 , J[6] = 0.700000  
 Node[7] = 12 , J[7] = 0.700000  
 Node[8] = 1 , J[8] = 0.666667  
 Node[9] = 14 , J[9] = 0.600000  
 Node[10] = 8 , J[10] = 0.600000  
 Node[11] = 19 , J[11] = 0.571429  
 Node[12] = 5 , J[12] = 0.571429  
 Node[13] = 25 , J[13] = 0.533333  
 Node[14] = 28 , J[14] = 0.533333  
 Node[15] = 20 , J[15] = 0.529412  
 Node[16] = 7 , J[16] = 0.500000  
 Node[17] = 3 , J[17] = 0.500000  
 Node[18] = 4 , J[18] = 0.500000  
 Node[19] = 18 , J[19] = 0.500000  
 Node[20] = 13 , J[20] = 0.416667  
 Node[21] = 2 , J[21] = 0.400000  
 Node[22] = 27 , J[22] = 0.357143  
 Node[23] = 6 , J[23] = 0.000000

**VERSION 2  
IDENTICAL**

I[8][7] = 0.636364 ; 7 / 11  
 I[21][20] = 0.571429 ; 4 / 7  
 I[22][20] = 0.571429 ; 4 / 7  
 I[22][21] = 0.571429 ; 4 / 7  
 I[23][20] = 0.571429 ; 4 / 7  
 I[23][21] = 0.571429 ; 4 / 7  
 I[23][22] = 0.571429 ; 4 / 7  
 I[20][19] = 0.428571 ; 3 / 7  
 I[21][19] = 0.428571 ; 3 / 7  
 I[22][19] = 0.428571 ; 3 / 7  
 I[23][19] = 0.428571 ; 3 / 7  
 I[16][15] = 0.400000 ; 2 / 5  
 I[17][15] = 0.400000 ; 2 / 5  
 I[17][16] = 0.400000 ; 2 / 5  
 I[18][15] = 0.400000 ; 2 / 5  
 I[18][16] = 0.400000 ; 2 / 5  
 I[18][17] = 0.400000 ; 2 / 5  
 I[30][6] = 0.333333 ; 2 / 6  
 I[4][3] = 0.300000 ; 3 / 10  
 I[15][14] = 0.200000 ; 1 / 5  
 I[16][14] = 0.200000 ; 1 / 5  
 I[17][14] = 0.200000 ; 1 / 5  
 I[18][14] = 0.200000 ; 1 / 5  
 I[13][5] = 0.153846 ; 2 / 13  
 I[5][3] = 0.133333 ; 2 / 15  
 I[5][4] = 0.133333 ; 2 / 15  
 I[13][2] = 0.111111 ; 1 / 9  
 I[30][2] = 0.111111 ; 1 / 9  
 I[30][3] = 0.111111 ; 1 / 9  
 I[30][4] = 0.100000 ; 1 / 10  
 I[3][2] = 0.090909 ; 1 / 11  
 I[4][2] = 0.090909 ; 1 / 11  
 I[6][2] = 0.090909 ; 1 / 11  
 I[6][3] = 0.090909 ; 1 / 11  
 I[13][3] = 0.090909 ; 1 / 11  
 I[13][4] = 0.090909 ; 1 / 11  
 I[6][4] = 0.083333 ; 1 / 12  
 I[7][1] = 0.076923 ; 1 / 13  
 I[8][1] = 0.076923 ; 1 / 13  
 I[30][5] = 0.071429 ; 1 / 14  
 I[5][2] = 0.066667 ; 1 / 15  
 I[6][5] = 0.062500 ; 1 / 16  
 JLL OTHER EDGES = 0.000000

Node[1] = 8 , I[1] = 0.636364  
 Node[2] = 7 , I[2] = 0.636364  
 Node[3] = 21 , I[3] = 0.571429  
 Node[4] = 20 , I[4] = 0.571429  
 Node[5] = 22 , I[5] = 0.571429  
 Node[6] = 23 , I[6] = 0.571429  
 Node[7] = 19 , I[7] = 0.428571  
 Node[8] = 16 , I[8] = 0.400000  
 Node[9] = 15 , I[9] = 0.400000  
 Node[10] = 17 , I[10] = 0.400000  
 Node[11] = 18 , I[11] = 0.400000  
 Node[12] = 30 , I[12] = 0.333333  
 Node[13] = 6 , I[13] = 0.333333  
 Node[14] = 4 , I[14] = 0.300000  
 Node[15] = 3 , I[15] = 0.300000  
 Node[16] = 14 , I[16] = 0.200000  
 Node[17] = 13 , I[17] = 0.153846  
 Node[18] = 5 , I[18] = 0.153846  
 Node[19] = 2 , I[19] = 0.111111  
 Node[20] = 1 , I[20] = 0.076923  
 Node[21] = 10 , I[21] = 0.000000  
 Node[22] = 11 , I[22] = 0.000000  
 Node[23] = 12 , I[23] = 0.000000  
 Node[24] = 25 , I[24] = 0.000000  
 Node[25] = 26 , I[25] = 0.000000  
 Node[26] = 0 , I[26] = 0.000000

**VERSION 2  
COINCIDENTAL**

C[19][13] = 0.500000 ; 4 / 8	C[19][18] = 0.250000 ; 2 / 8
C[20][13] = 0.500000 ; 4 / 8	C[20][14] = 0.250000 ; 2 / 8
C[21][13] = 0.500000 ; 4 / 8	C[20][15] = 0.250000 ; 2 / 8
C[22][13] = 0.500000 ; 4 / 8	C[20][16] = 0.250000 ; 2 / 8
C[23][13] = 0.500000 ; 4 / 8	C[20][17] = 0.250000 ; 2 / 8
C[14][13] = 0.428571 ; 3 / 7	C[20][18] = 0.250000 ; 2 / 8
C[15][13] = 0.428571 ; 3 / 7	C[21][14] = 0.250000 ; 2 / 8
C[16][13] = 0.428571 ; 3 / 7	C[21][15] = 0.250000 ; 2 / 8
C[17][13] = 0.428571 ; 3 / 7	C[21][16] = 0.250000 ; 2 / 8
C[18][13] = 0.428571 ; 3 / 7	C[21][17] = 0.250000 ; 2 / 8
C[15][14] = 0.400000 ; 2 / 5	C[21][18] = 0.250000 ; 2 / 8
C[16][14] = 0.400000 ; 2 / 5	C[22][14] = 0.250000 ; 2 / 8
C[17][14] = 0.400000 ; 2 / 5	C[22][15] = 0.250000 ; 2 / 8
C[18][14] = 0.400000 ; 2 / 5	C[22][16] = 0.250000 ; 2 / 8
C[8][7] = 0.363636 ; 4 / 11	C[22][17] = 0.250000 ; 2 / 8
C[25][5] = 0.352941 ; 6 / 17	C[22][18] = 0.250000 ; 2 / 8
C[5][1] = 0.333333 ; 4 / 12	C[23][14] = 0.250000 ; 2 / 8
C[19][2] = 0.333333 ; 3 / 9	C[23][15] = 0.250000 ; 2 / 8
C[20][2] = 0.333333 ; 3 / 9	C[23][16] = 0.250000 ; 2 / 8
C[21][2] = 0.333333 ; 3 / 9	C[23][17] = 0.250000 ; 2 / 8
C[22][2] = 0.333333 ; 3 / 9	C[23][18] = 0.250000 ; 2 / 8
C[23][2] = 0.333333 ; 3 / 9	C[26][3] = 0.250000 ; 2 / 8
C[30][6] = 0.333333 ; 2 / 6	C[26][5] = 0.250000 ; 3 / 12
C[22][5] = 0.307692 ; 4 / 13	C[26][13] = 0.250000 ; 2 / 8
C[25][1] = 0.307692 ; 4 / 13	C[26][19] = 0.250000 ; 2 / 8
C[14][1] = 0.285714 ; 2 / 7	C[26][20] = 0.250000 ; 2 / 8
C[15][1] = 0.285714 ; 2 / 7	C[26][21] = 0.250000 ; 2 / 8
C[16][1] = 0.285714 ; 2 / 7	C[26][22] = 0.250000 ; 2 / 8
C[17][1] = 0.285714 ; 2 / 7	C[26][23] = 0.250000 ; 2 / 8
C[18][1] = 0.285714 ; 2 / 7	C[7][5] = 0.222222 ; 4 / 18
C[20][19] = 0.285714 ; 2 / 7	C[8][5] = 0.222222 ; 4 / 18
C[21][19] = 0.285714 ; 2 / 7	C[13][1] = 0.222222 ; 2 / 9
C[22][19] = 0.285714 ; 2 / 7	C[13][2] = 0.222222 ; 2 / 9
C[23][19] = 0.285714 ; 2 / 7	C[26][4] = 0.222222 ; 2 / 9
C[25][7] = 0.277778 ; 5 / 18	C[19][5] = 0.214286 ; 3 / 14
C[25][8] = 0.277778 ; 5 / 18	C[20][5] = 0.214286 ; 3 / 14
C[14][2] = 0.250000 ; 2 / 8	C[21][5] = 0.214286 ; 3 / 14
C[15][2] = 0.250000 ; 2 / 8	C[23][5] = 0.214286 ; 3 / 14
C[16][2] = 0.250000 ; 2 / 8	C[4][1] = 0.200000 ; 2 / 10
C[17][2] = 0.250000 ; 2 / 8	C[16][15] = 0.200000 ; 1 / 5
C[17][5] = 0.250000 ; 3 / 12	C[17][15] = 0.200000 ; 1 / 5
C[18][2] = 0.250000 ; 2 / 8	C[17][16] = 0.200000 ; 1 / 5
C[19][14] = 0.250000 ; 2 / 8	C[18][15] = 0.200000 ; 1 / 5
C[19][15] = 0.250000 ; 2 / 8	C[18][16] = 0.200000 ; 1 / 5
C[19][16] = 0.250000 ; 2 / 8	C[18][17] = 0.200000 ; 1 / 5
C[19][17] = 0.250000 ; 2 / 8	C[10][6] = 0.166667 ; 1 / 6
	C[11][6] = 0.166667 ; 1 / 6
	C[7][1] = 0.153846 ; 2 / 13
	C[8][1] = 0.153846 ; 2 / 13
	C[13][5] = 0.153846 ; 2 / 13
	C[14][5] = 0.153846 ; 2 / 13



C[14][7] = 0.153846 ; 2 / 13  
 C[14][8] = 0.153846 ; 2 / 13  
 C[15][5] = 0.153846 ; 2 / 13  
 C[15][7] = 0.153846 ; 2 / 13  
 C[15][8] = 0.153846 ; 2 / 13  
 C[16][5] = 0.153846 ; 2 / 13  
 C[16][7] = 0.153846 ; 2 / 13  
 C[16][8] = 0.153846 ; 2 / 13  
 C[17][7] = 0.153846 ; 2 / 13  
 C[17][8] = 0.153846 ; 2 / 13  
 C[18][5] = 0.153846 ; 2 / 13  
 C[18][7] = 0.153846 ; 2 / 13  
 C[18][8] = 0.153846 ; 2 / 13  
 C[30][7] = 0.153846 ; 2 / 13  
 C[30][8] = 0.153846 ; 2 / 13  
 C[21][20] = 0.142857 ; 1 / 7  
 C[22][20] = 0.142857 ; 1 / 7  
 C[22][21] = 0.142857 ; 1 / 7  
 C[23][20] = 0.142857 ; 1 / 7  
 C[23][21] = 0.142857 ; 1 / 7  
 C[23][22] = 0.142857 ; 1 / 7  
 C[25][14] = 0.142857 ; 2 / 14  
 C[25][15] = 0.142857 ; 2 / 14  
 C[25][16] = 0.142857 ; 2 / 14  
 C[25][17] = 0.142857 ; 2 / 14  
 C[25][18] = 0.142857 ; 2 / 14  
 C[30][14] = 0.142857 ; 1 / 7  
 C[30][15] = 0.142857 ; 1 / 7  
 C[30][16] = 0.142857 ; 1 / 7  
 C[30][17] = 0.142857 ; 1 / 7  
 C[30][18] = 0.142857 ; 1 / 7  
 C[30][25] = 0.142857 ; 2 / 14  
 C[30][26] = 0.142857 ; 1 / 7  
 C[7][6] = 0.133333 ; 2 / 15  
 C[8][6] = 0.133333 ; 2 / 15  
 C[13][7] = 0.133333 ; 2 / 15  
 C[13][8] = 0.133333 ; 2 / 15  
 C[7][4] = 0.125000 ; 2 / 16  
 C[8][4] = 0.125000 ; 2 / 16  
 C[25][6] = 0.125000 ; 2 / 16  
 C[25][13] = 0.125000 ; 2 / 16  
 C[26][1] = 0.125000 ; 1 / 8  
 C[30][1] = 0.125000 ; 1 / 8  
 C[25][4] = 0.117647 ; 2 / 17  
 C[14][6] = 0.111111 ; 1 / 9  
 C[15][6] = 0.111111 ; 1 / 9  
 C[16][6] = 0.111111 ; 1 / 9  
 C[17][6] = 0.111111 ; 1 / 9  
 C[18][6] = 0.111111 ; 1 / 9  
 C[26][2] = 0.111111 ; 1 / 9  
 C[26][6] = 0.111111 ; 1 / 9

C[30][13] = 0.111111 ; 1 / 9  
 C[30][19] = 0.111111 ; 1 / 9  
 C[30][20] = 0.111111 ; 1 / 9  
 C[30][21] = 0.111111 ; 1 / 9  
 C[30][22] = 0.111111 ; 1 / 9  
 C[30][23] = 0.111111 ; 1 / 9  
 C[2][1] = 0.100000 ; 1 / 10  
 C[3][1] = 0.100000 ; 1 / 10  
 C[6][1] = 0.100000 ; 1 / 10  
 C[14][4] = 0.100000 ; 1 / 10  
 C[15][4] = 0.100000 ; 1 / 10  
 C[16][4] = 0.100000 ; 1 / 10  
 C[17][4] = 0.100000 ; 1 / 10  
 C[18][4] = 0.100000 ; 1 / 10  
 C[19][1] = 0.100000 ; 1 / 10  
 C[20][1] = 0.100000 ; 1 / 10  
 C[21][1] = 0.100000 ; 1 / 10  
 C[22][1] = 0.100000 ; 1 / 10  
 C[23][1] = 0.100000 ; 1 / 10  
 C[4][2] = 0.090909 ; 1 / 11  
 C[13][4] = 0.090909 ; 1 / 11  
 C[13][6] = 0.090909 ; 1 / 11  
 C[19][3] = 0.090909 ; 1 / 11  
 C[19][6] = 0.090909 ; 1 / 11  
 C[20][3] = 0.090909 ; 1 / 11  
 C[20][6] = 0.090909 ; 1 / 11  
 C[21][3] = 0.090909 ; 1 / 11  
 C[21][6] = 0.090909 ; 1 / 11  
 C[22][3] = 0.090909 ; 1 / 11  
 C[22][6] = 0.090909 ; 1 / 11  
 C[23][3] = 0.090909 ; 1 / 11  
 C[23][6] = 0.090909 ; 1 / 11  
 C[19][4] = 0.083333 ; 1 / 12  
 C[20][4] = 0.083333 ; 1 / 12  
 C[21][4] = 0.083333 ; 1 / 12  
 C[22][4] = 0.083333 ; 1 / 12  
 C[23][4] = 0.083333 ; 1 / 12  
 C[26][7] = 0.071429 ; 1 / 14  
 C[26][8] = 0.071429 ; 1 / 14  
 C[5][2] = 0.066667 ; 1 / 15  
 C[5][4] = 0.066667 ; 1 / 15  
 C[26][25] = 0.066667 ; 1 / 15  
 C[7][2] = 0.062500 ; 1 / 16  
 C[7][3] = 0.062500 ; 1 / 16  
 C[8][2] = 0.062500 ; 1 / 16  
 C[8][3] = 0.062500 ; 1 / 16  
 C[19][7] = 0.062500 ; 1 / 16  
 C[19][8] = 0.062500 ; 1 / 16  
 C[20][7] = 0.062500 ; 1 / 16  
 C[20][8] = 0.062500 ; 1 / 16  
 C[21][7] = 0.062500 ; 1 / 16

```

C[21][8] = 0.062500 ; 1 / 16
C[22][7] = 0.062500 ; 1 / 16
C[22][8] = 0.062500 ; 1 / 16
C[23][7] = 0.062500 ; 1 / 16
C[23][8] = 0.062500 ; 1 / 16
C[25][2] = 0.058824 ; 1 / 17
C[25][3] = 0.058824 ; 1 / 17
C[25][19] = 0.058824 ; 1 / 17
C[25][20] = 0.058824 ; 1 / 17
C[25][21] = 0.058824 ; 1 / 17
C[25][22] = 0.058824 ; 1 / 17
C[25][23] = 0.058824 ; 1 / 17
JLL OTHER EDGES = 0.000000

```

```

Node[1] = 19 , C[1] = 0.500000
Node[2] = 13 , C[2] = 0.500000
Node[3] = 20 , C[3] = 0.500000
Node[4] = 21 , C[4] = 0.500000
Node[5] = 22 , C[5] = 0.500000
Node[6] = 23 , C[6] = 0.500000
Node[7] = 14 , C[7] = 0.428571
Node[8] = 15 , C[8] = 0.428571
Node[9] = 16 , C[9] = 0.428571
Node[10] = 17 , C[10] = 0.428571
Node[11] = 18 , C[11] = 0.428571
Node[12] = 8 , C[12] = 0.363636
Node[13] = 7 , C[13] = 0.363636
Node[14] = 25 , C[14] = 0.352941
Node[15] = 5 , C[15] = 0.352941
Node[16] = 1 , C[16] = 0.333333
Node[17] = 2 , C[17] = 0.333333
Node[18] = 30 , C[18] = 0.333333
Node[19] = 6 , C[19] = 0.333333
Node[20] = 26 , C[20] = 0.250000
Node[21] = 3 , C[21] = 0.250000
Node[22] = 4 , C[22] = 0.222222
Node[23] = 10 , C[23] = 0.166667
Node[24] = 11 , C[24] = 0.166667
Node[25] = 12 , C[25] = 0.000000

```

**VERSION 2  
COMPOSITE**

J[8][7] = 1.000000 ; 11 / 11	J[16][1] = 0.285714 ; 2 / 7
J[20][19] = 0.714286 ; 5 / 7	J[17][1] = 0.285714 ; 2 / 7
J[21][19] = 0.714286 ; 5 / 7	J[18][1] = 0.285714 ; 2 / 7
J[21][20] = 0.714286 ; 5 / 7	J[25][7] = 0.277778 ; 5 / 18
J[22][19] = 0.714286 ; 5 / 7	J[25][8] = 0.277778 ; 5 / 18
J[22][20] = 0.714286 ; 5 / 7	J[14][2] = 0.250000 ; 2 / 8
J[22][21] = 0.714286 ; 5 / 7	J[15][2] = 0.250000 ; 2 / 8
J[23][19] = 0.714286 ; 5 / 7	J[16][2] = 0.250000 ; 2 / 8
J[23][20] = 0.714286 ; 5 / 7	J[17][2] = 0.250000 ; 2 / 8
J[23][21] = 0.714286 ; 5 / 7	J[17][5] = 0.250000 ; 3 / 12
J[23][22] = 0.714286 ; 5 / 7	J[18][2] = 0.250000 ; 2 / 8
J[30][6] = 0.666667 ; 4 / 6	J[19][14] = 0.250000 ; 2 / 8
J[15][14] = 0.600000 ; 3 / 5	J[19][15] = 0.250000 ; 2 / 8
J[16][14] = 0.600000 ; 3 / 5	J[19][16] = 0.250000 ; 2 / 8
J[16][15] = 0.600000 ; 3 / 5	J[19][17] = 0.250000 ; 2 / 8
J[17][14] = 0.600000 ; 3 / 5	J[19][18] = 0.250000 ; 2 / 8
J[17][15] = 0.600000 ; 3 / 5	J[20][14] = 0.250000 ; 2 / 8
J[17][16] = 0.600000 ; 3 / 5	J[20][15] = 0.250000 ; 2 / 8
J[18][14] = 0.600000 ; 3 / 5	J[20][16] = 0.250000 ; 2 / 8
J[18][15] = 0.600000 ; 3 / 5	J[20][17] = 0.250000 ; 2 / 8
J[18][16] = 0.600000 ; 3 / 5	J[20][18] = 0.250000 ; 2 / 8
J[18][17] = 0.600000 ; 3 / 5	J[21][14] = 0.250000 ; 2 / 8
J[19][13] = 0.500000 ; 4 / 8	J[21][15] = 0.250000 ; 2 / 8
J[20][13] = 0.500000 ; 4 / 8	J[21][16] = 0.250000 ; 2 / 8
J[21][13] = 0.500000 ; 4 / 8	J[21][17] = 0.250000 ; 2 / 8
J[22][13] = 0.500000 ; 4 / 8	J[21][18] = 0.250000 ; 2 / 8
J[23][13] = 0.500000 ; 4 / 8	J[22][14] = 0.250000 ; 2 / 8
J[14][13] = 0.428571 ; 3 / 7	J[22][15] = 0.250000 ; 2 / 8
J[15][13] = 0.428571 ; 3 / 7	J[22][16] = 0.250000 ; 2 / 8
J[16][13] = 0.428571 ; 3 / 7	J[22][17] = 0.250000 ; 2 / 8
J[17][13] = 0.428571 ; 3 / 7	J[22][18] = 0.250000 ; 2 / 8
J[18][13] = 0.428571 ; 3 / 7	J[23][14] = 0.250000 ; 2 / 8
J[25][5] = 0.352941 ; 6 / 17	J[23][15] = 0.250000 ; 2 / 8
J[5][1] = 0.333333 ; 4 / 12	J[23][16] = 0.250000 ; 2 / 8
J[13][2] = 0.333333 ; 3 / 9	J[23][17] = 0.250000 ; 2 / 8
J[19][2] = 0.333333 ; 3 / 9	J[23][18] = 0.250000 ; 2 / 8
J[20][2] = 0.333333 ; 3 / 9	J[26][3] = 0.250000 ; 2 / 8
J[21][2] = 0.333333 ; 3 / 9	J[26][5] = 0.250000 ; 3 / 12
J[22][2] = 0.333333 ; 3 / 9	J[26][13] = 0.250000 ; 2 / 8
J[23][2] = 0.333333 ; 3 / 9	J[26][19] = 0.250000 ; 2 / 8
J[13][5] = 0.307692 ; 4 / 13	J[26][20] = 0.250000 ; 2 / 8
J[22][5] = 0.307692 ; 4 / 13	J[26][21] = 0.250000 ; 2 / 8
J[25][1] = 0.307692 ; 4 / 13	J[26][22] = 0.250000 ; 2 / 8
J[4][3] = 0.300000 ; 3 / 10	J[26][23] = 0.250000 ; 2 / 8
J[14][1] = 0.285714 ; 2 / 7	J[7][1] = 0.230769 ; 3 / 13
J[15][1] = 0.285714 ; 2 / 7	J[8][1] = 0.230769 ; 3 / 13
	J[7][5] = 0.222222 ; 4 / 18
	J[8][5] = 0.222222 ; 4 / 18
	J[13][1] = 0.222222 ; 2 / 9
	J[26][4] = 0.222222 ; 2 / 9
	J[19][5] = 0.214286 ; 3 / 14

J[20][5] = 0.214286 ; 3 / 14  
 J[21][5] = 0.214286 ; 3 / 14  
 J[23][5] = 0.214286 ; 3 / 14  
 J[4][1] = 0.200000 ; 2 / 10  
 J[5][4] = 0.200000 ; 3 / 15  
 J[4][2] = 0.181818 ; 2 / 11  
 J[13][4] = 0.181818 ; 2 / 11  
 J[10][6] = 0.166667 ; 1 / 6  
 J[11][6] = 0.166667 ; 1 / 6  
 J[14][5] = 0.153846 ; 2 / 13  
 J[14][7] = 0.153846 ; 2 / 13  
 J[14][8] = 0.153846 ; 2 / 13  
 J[15][5] = 0.153846 ; 2 / 13  
 J[15][7] = 0.153846 ; 2 / 13  
 J[15][8] = 0.153846 ; 2 / 13  
 J[16][5] = 0.153846 ; 2 / 13  
 J[16][7] = 0.153846 ; 2 / 13  
 J[16][8] = 0.153846 ; 2 / 13  
 J[17][7] = 0.153846 ; 2 / 13  
 J[17][8] = 0.153846 ; 2 / 13  
 J[18][5] = 0.153846 ; 2 / 13  
 J[18][7] = 0.153846 ; 2 / 13  
 J[18][8] = 0.153846 ; 2 / 13  
 J[30][7] = 0.153846 ; 2 / 13  
 J[30][8] = 0.153846 ; 2 / 13  
 J[25][14] = 0.142857 ; 2 / 14  
 J[25][15] = 0.142857 ; 2 / 14  
 J[25][16] = 0.142857 ; 2 / 14  
 J[25][17] = 0.142857 ; 2 / 14  
 J[25][18] = 0.142857 ; 2 / 14  
 J[30][14] = 0.142857 ; 1 / 7  
 J[30][15] = 0.142857 ; 1 / 7  
 J[30][16] = 0.142857 ; 1 / 7  
 J[30][17] = 0.142857 ; 1 / 7  
 J[30][18] = 0.142857 ; 1 / 7  
 J[30][25] = 0.142857 ; 2 / 14  
 J[30][26] = 0.142857 ; 1 / 7  
 J[5][2] = 0.133333 ; 2 / 15  
 J[5][3] = 0.133333 ; 2 / 15  
 J[7][6] = 0.133333 ; 2 / 15  
 J[8][6] = 0.133333 ; 2 / 15  
 J[13][7] = 0.133333 ; 2 / 15  
 J[13][8] = 0.133333 ; 2 / 15  
 J[7][4] = 0.125000 ; 2 / 16  
 J[8][4] = 0.125000 ; 2 / 16  
 J[25][6] = 0.125000 ; 2 / 16  
 J[25][13] = 0.125000 ; 2 / 16  
 J[26][1] = 0.125000 ; 1 / 8  
 J[30][1] = 0.125000 ; 1 / 8  
 J[25][4] = 0.117647 ; 2 / 17  
 J[14][6] = 0.111111 ; 1 / 9

J[15][6] = 0.111111 ; 1 / 9  
 J[16][6] = 0.111111 ; 1 / 9  
 J[17][6] = 0.111111 ; 1 / 9  
 J[18][6] = 0.111111 ; 1 / 9  
 J[26][2] = 0.111111 ; 1 / 9  
 J[26][6] = 0.111111 ; 1 / 9  
 J[30][2] = 0.111111 ; 1 / 9  
 J[30][3] = 0.111111 ; 1 / 9  
 J[30][13] = 0.111111 ; 1 / 9  
 J[30][19] = 0.111111 ; 1 / 9  
 J[30][20] = 0.111111 ; 1 / 9  
 J[30][21] = 0.111111 ; 1 / 9  
 J[30][22] = 0.111111 ; 1 / 9  
 J[30][23] = 0.111111 ; 1 / 9  
 J[2][1] = 0.100000 ; 1 / 10  
 J[3][1] = 0.100000 ; 1 / 10  
 J[6][1] = 0.100000 ; 1 / 10  
 J[14][4] = 0.100000 ; 1 / 10  
 J[15][4] = 0.100000 ; 1 / 10  
 J[16][4] = 0.100000 ; 1 / 10  
 J[17][4] = 0.100000 ; 1 / 10  
 J[18][4] = 0.100000 ; 1 / 10  
 J[19][1] = 0.100000 ; 1 / 10  
 J[20][1] = 0.100000 ; 1 / 10  
 J[21][1] = 0.100000 ; 1 / 10  
 J[22][1] = 0.100000 ; 1 / 10  
 J[23][1] = 0.100000 ; 1 / 10  
 J[30][4] = 0.100000 ; 1 / 10  
 J[3][2] = 0.090909 ; 1 / 11  
 J[6][2] = 0.090909 ; 1 / 11  
 J[6][3] = 0.090909 ; 1 / 11  
 J[13][3] = 0.090909 ; 1 / 11  
 J[13][6] = 0.090909 ; 1 / 11  
 J[19][3] = 0.090909 ; 1 / 11  
 J[19][6] = 0.090909 ; 1 / 11  
 J[20][3] = 0.090909 ; 1 / 11  
 J[20][6] = 0.090909 ; 1 / 11  
 J[21][3] = 0.090909 ; 1 / 11  
 J[21][6] = 0.090909 ; 1 / 11  
 J[22][3] = 0.090909 ; 1 / 11  
 J[22][6] = 0.090909 ; 1 / 11  
 J[23][3] = 0.090909 ; 1 / 11  
 J[23][6] = 0.090909 ; 1 / 11  
 J[6][4] = 0.083333 ; 1 / 12  
 J[19][4] = 0.083333 ; 1 / 12  
 J[20][4] = 0.083333 ; 1 / 12  
 J[21][4] = 0.083333 ; 1 / 12  
 J[22][4] = 0.083333 ; 1 / 12  
 J[23][4] = 0.083333 ; 1 / 12  
 J[26][7] = 0.071429 ; 1 / 14  
 J[26][8] = 0.071429 ; 1 / 14

J[30][5] = 0.071429 ; 1 / 14  
 J[26][25] = 0.066667 ; 1 / 15  
 J[6][5] = 0.062500 ; 1 / 16  
 J[7][2] = 0.062500 ; 1 / 16  
 J[7][3] = 0.062500 ; 1 / 16  
 J[8][2] = 0.062500 ; 1 / 16  
 J[8][3] = 0.062500 ; 1 / 16  
 J[19][7] = 0.062500 ; 1 / 16  
 J[19][8] = 0.062500 ; 1 / 16  
 J[20][7] = 0.062500 ; 1 / 16  
 J[20][8] = 0.062500 ; 1 / 16  
 J[21][7] = 0.062500 ; 1 / 16  
 J[21][8] = 0.062500 ; 1 / 16  
 J[22][7] = 0.062500 ; 1 / 16  
 J[22][8] = 0.062500 ; 1 / 16  
 J[23][7] = 0.062500 ; 1 / 16  
 J[23][8] = 0.062500 ; 1 / 16  
 J[25][2] = 0.058824 ; 1 / 17  
 J[25][3] = 0.058824 ; 1 / 17  
 J[25][19] = 0.058824 ; 1 / 17  
 J[25][20] = 0.058824 ; 1 / 17  
 J[25][21] = 0.058824 ; 1 / 17  
 J[25][22] = 0.058824 ; 1 / 17  
 J[25][23] = 0.058824 ; 1 / 17  
 JLL OTHER EDGES = 0.000000

Node[1] = 8 , J[1] = 1.000000  
 Node[2] = 7 , J[2] = 1.000000  
 Node[3] = 20 , J[3] = 0.714286  
 Node[4] = 19 , J[4] = 0.714286  
 Node[5] = 21 , J[5] = 0.714286  
 Node[6] = 22 , J[6] = 0.714286  
 Node[7] = 23 , J[7] = 0.714286  
 Node[8] = 30 , J[8] = 0.666667  
 Node[9] = 6 , J[9] = 0.666667  
 Node[10] = 15 , J[10] = 0.600000  
 Node[11] = 14 , J[11] = 0.600000  
 Node[12] = 16 , J[12] = 0.600000  
 Node[13] = 17 , J[13] = 0.600000  
 Node[14] = 18 , J[14] = 0.600000  
 Node[15] = 13 , J[15] = 0.500000  
 Node[16] = 25 , J[16] = 0.352941  
 Node[17] = 5 , J[17] = 0.352941  
 Node[18] = 1 , J[18] = 0.333333  
 Node[19] = 2 , J[19] = 0.333333  
 Node[20] = 4 , J[20] = 0.300000  
 Node[21] = 3 , J[21] = 0.300000  
 Node[22] = 26 , J[22] = 0.250000  
 Node[23] = 10 , J[23] = 0.166667  
 Node[24] = 11 , J[24] = 0.166667  
 Node[25] = 12 , J[25] = 0.000000

**VERSION 3**  
**IDENTICAL**

I[44][18] = 0.909091 ; 10 / 11	I[25][3] = 0.250000 ; 2 / 8
I[44][7] = 0.615385 ; 8 / 13	I[25][6] = 0.250000 ; 1 / 4
I[2][1] = 0.600000 ; 3 / 5	I[38][5] = 0.250000 ; 2 / 8
I[18][7] = 0.571429 ; 8 / 14	I[39][26] = 0.250000 ; 2 / 8
I[33][32] = 0.500000 ; 2 / 4	I[40][26] = 0.250000 ; 2 / 8
I[34][32] = 0.500000 ; 2 / 4	I[41][26] = 0.250000 ; 2 / 8
I[35][32] = 0.500000 ; 2 / 4	I[42][26] = 0.250000 ; 2 / 8
I[36][32] = 0.500000 ; 2 / 4	I[38][28] = 0.222222 ; 2 / 9
I[37][32] = 0.500000 ; 2 / 4	I[39][5] = 0.222222 ; 2 / 9
I[40][39] = 0.500000 ; 3 / 6	I[40][5] = 0.222222 ; 2 / 9
I[41][39] = 0.500000 ; 3 / 6	I[41][5] = 0.222222 ; 2 / 9
I[41][40] = 0.500000 ; 3 / 6	I[42][5] = 0.222222 ; 2 / 9
I[42][39] = 0.500000 ; 3 / 6	I[6][2] = 0.200000 ; 1 / 5
I[42][40] = 0.500000 ; 3 / 6	I[22][6] = 0.200000 ; 1 / 5
I[42][41] = 0.500000 ; 3 / 6	I[23][2] = 0.200000 ; 2 / 10
I[45][7] = 0.470588 ; 8 / 17	I[38][10] = 0.200000 ; 1 / 5
I[45][18] = 0.470588 ; 8 / 17	I[39][10] = 0.200000 ; 1 / 5
I[45][44] = 0.470588 ; 8 / 17	I[40][10] = 0.200000 ; 1 / 5
I[18][9] = 0.428571 ; 9 / 21	I[41][10] = 0.200000 ; 1 / 5
I[26][5] = 0.428571 ; 3 / 7	I[42][10] = 0.200000 ; 1 / 5
I[38][26] = 0.428571 ; 3 / 7	I[23][1] = 0.181818 ; 2 / 11
I[34][33] = 0.400000 ; 2 / 5	I[24][23] = 0.181818 ; 2 / 11
I[35][33] = 0.400000 ; 2 / 5	I[10][5] = 0.166667 ; 1 / 6
I[35][34] = 0.400000 ; 2 / 5	I[22][2] = 0.166667 ; 1 / 6
I[36][33] = 0.400000 ; 2 / 5	I[23][12] = 0.166667 ; 2 / 12
I[36][34] = 0.400000 ; 2 / 5	I[24][2] = 0.166667 ; 1 / 6
I[36][35] = 0.400000 ; 2 / 5	I[24][3] = 0.166667 ; 1 / 6
I[37][33] = 0.400000 ; 2 / 5	I[25][2] = 0.166667 ; 1 / 6
I[37][34] = 0.400000 ; 2 / 5	I[25][22] = 0.166667 ; 1 / 6
I[37][35] = 0.400000 ; 2 / 5	I[26][10] = 0.166667 ; 1 / 6
I[37][36] = 0.400000 ; 2 / 5	I[32][3] = 0.166667 ; 1 / 6
I[9][7] = 0.380952 ; 8 / 21	I[32][26] = 0.166667 ; 1 / 6
I[44][9] = 0.380952 ; 8 / 21	I[36][3] = 0.166667 ; 1 / 6
I[45][9] = 0.380952 ; 8 / 21	I[38][4] = 0.166667 ; 1 / 6
I[16][11] = 0.333333 ; 2 / 6	I[22][18] = 0.153846 ; 2 / 13
I[26][24] = 0.333333 ; 2 / 6	I[6][1] = 0.142857 ; 1 / 7
I[28][26] = 0.333333 ; 3 / 9	I[24][1] = 0.142857 ; 1 / 7
I[38][24] = 0.333333 ; 2 / 6	I[24][5] = 0.142857 ; 1 / 7
I[39][38] = 0.333333 ; 2 / 6	I[33][3] = 0.142857 ; 1 / 7
I[40][38] = 0.333333 ; 2 / 6	I[33][26] = 0.142857 ; 1 / 7
I[41][38] = 0.333333 ; 2 / 6	I[34][3] = 0.142857 ; 1 / 7
I[42][38] = 0.333333 ; 2 / 6	I[34][26] = 0.142857 ; 1 / 7
I[3][2] = 0.285714 ; 2 / 7	I[35][3] = 0.142857 ; 1 / 7
I[15][3] = 0.272727 ; 3 / 11	I[35][26] = 0.142857 ; 1 / 7
I[3][1] = 0.250000 ; 2 / 8	I[36][26] = 0.142857 ; 1 / 7
I[24][10] = 0.250000 ; 1 / 4	I[37][3] = 0.142857 ; 1 / 7
	I[37][26] = 0.142857 ; 1 / 7
	I[39][4] = 0.142857 ; 1 / 7
	I[39][24] = 0.142857 ; 1 / 7
	I[40][4] = 0.142857 ; 1 / 7
	I[40][24] = 0.142857 ; 1 / 7

I[41][4] = 0.142857 ; 1 / 7  
 I[41][24] = 0.142857 ; 1 / 7  
 I[42][4] = 0.142857 ; 1 / 7  
 I[42][24] = 0.142857 ; 1 / 7  
 I[43][6] = 0.142857 ; 1 / 7  
 I[43][18] = 0.142857 ; 2 / 14  
 I[44][43] = 0.142857 ; 2 / 14  
 I[11][2] = 0.125000 ; 1 / 8  
 I[11][6] = 0.125000 ; 1 / 8  
 I[12][2] = 0.125000 ; 1 / 8  
 I[12][7] = 0.125000 ; 2 / 16  
 I[16][2] = 0.125000 ; 1 / 8  
 I[16][6] = 0.125000 ; 1 / 8  
 I[22][1] = 0.125000 ; 1 / 8  
 I[25][1] = 0.125000 ; 1 / 8  
 I[26][3] = 0.125000 ; 1 / 8  
 I[26][4] = 0.125000 ; 1 / 8  
 I[28][2] = 0.125000 ; 1 / 8  
 I[28][6] = 0.125000 ; 1 / 8  
 I[28][24] = 0.125000 ; 1 / 8  
 I[43][25] = 0.125000 ; 1 / 8  
 I[11][1] = 0.111111 ; 1 / 9  
 I[12][1] = 0.111111 ; 1 / 9  
 I[12][6] = 0.111111 ; 1 / 9  
 I[12][11] = 0.111111 ; 1 / 9  
 I[16][1] = 0.111111 ; 1 / 9  
 I[16][12] = 0.111111 ; 1 / 9  
 I[22][11] = 0.111111 ; 1 / 9  
 I[22][16] = 0.111111 ; 1 / 9  
 I[23][10] = 0.111111 ; 1 / 9  
 I[25][11] = 0.111111 ; 1 / 9  
 I[25][16] = 0.111111 ; 1 / 9  
 I[28][1] = 0.111111 ; 1 / 9  
 I[28][5] = 0.111111 ; 1 / 9  
 I[28][12] = 0.111111 ; 1 / 9  
 I[28][22] = 0.111111 ; 1 / 9  
 I[28][25] = 0.111111 ; 1 / 9  
 I[43][2] = 0.111111 ; 1 / 9  
 I[43][22] = 0.111111 ; 1 / 9  
 I[45][38] = 0.111111 ; 2 / 18  
 I[15][4] = 0.100000 ; 1 / 10  
 I[22][12] = 0.100000 ; 1 / 10  
 I[24][15] = 0.100000 ; 1 / 10  
 I[25][12] = 0.100000 ; 1 / 10  
 I[28][4] = 0.100000 ; 1 / 10  
 I[28][11] = 0.100000 ; 1 / 10  
 I[28][16] = 0.100000 ; 1 / 10  
 I[32][15] = 0.100000 ; 1 / 10  
 I[39][28] = 0.100000 ; 1 / 10  
 I[40][28] = 0.100000 ; 1 / 10  
 I[41][28] = 0.100000 ; 1 / 10

I[42][28] = 0.100000 ; 1 / 10  
 I[18][6] = 0.090909 ; 1 / 11  
 I[23][6] = 0.090909 ; 1 / 11  
 I[25][15] = 0.090909 ; 1 / 11  
 I[26][15] = 0.090909 ; 1 / 11  
 I[33][15] = 0.090909 ; 1 / 11  
 I[34][15] = 0.090909 ; 1 / 11  
 I[35][15] = 0.090909 ; 1 / 11  
 I[36][15] = 0.090909 ; 1 / 11  
 I[37][15] = 0.090909 ; 1 / 11  
 I[38][23] = 0.090909 ; 1 / 11  
 I[43][1] = 0.090909 ; 1 / 11  
 I[43][39] = 0.090909 ; 1 / 11  
 I[43][40] = 0.090909 ; 1 / 11  
 I[43][41] = 0.090909 ; 1 / 11  
 I[43][42] = 0.090909 ; 1 / 11  
 I[44][6] = 0.090909 ; 1 / 11  
 I[12][9] = 0.086957 ; 2 / 23  
 I[15][2] = 0.083333 ; 1 / 12  
 I[23][5] = 0.083333 ; 1 / 12  
 I[23][22] = 0.083333 ; 1 / 12  
 I[25][18] = 0.083333 ; 1 / 12  
 I[25][23] = 0.083333 ; 1 / 12  
 I[26][23] = 0.083333 ; 1 / 12  
 I[28][23] = 0.083333 ; 1 / 12  
 I[39][23] = 0.083333 ; 1 / 12  
 I[40][23] = 0.083333 ; 1 / 12  
 I[41][23] = 0.083333 ; 1 / 12  
 I[42][23] = 0.083333 ; 1 / 12  
 I[43][4] = 0.083333 ; 1 / 12  
 I[43][11] = 0.083333 ; 1 / 12  
 I[43][16] = 0.083333 ; 1 / 12  
 I[43][28] = 0.083333 ; 1 / 12  
 I[44][2] = 0.083333 ; 1 / 12  
 I[44][25] = 0.083333 ; 1 / 12  
 I[15][1] = 0.076923 ; 1 / 13  
 I[18][2] = 0.076923 ; 1 / 13  
 I[22][15] = 0.076923 ; 1 / 13  
 I[23][3] = 0.076923 ; 1 / 13  
 I[23][11] = 0.076923 ; 1 / 13  
 I[23][16] = 0.076923 ; 1 / 13  
 I[24][18] = 0.076923 ; 1 / 13  
 I[43][12] = 0.076923 ; 1 / 13  
 I[44][22] = 0.076923 ; 1 / 13  
 I[44][24] = 0.076923 ; 1 / 13  
 I[7][2] = 0.071429 ; 1 / 14  
 I[7][6] = 0.071429 ; 1 / 14  
 I[44][1] = 0.071429 ; 1 / 14  
 I[44][38] = 0.071429 ; 1 / 14  
 I[7][1] = 0.066667 ; 1 / 15  
 I[15][11] = 0.066667 ; 1 / 15

I[16][15] = 0.066667 ; 1 / 15  
 I[18][1] = 0.066667 ; 1 / 15  
 I[25][7] = 0.066667 ; 1 / 15  
 I[38][18] = 0.066667 ; 1 / 15  
 I[43][23] = 0.066667 ; 1 / 15  
 I[44][11] = 0.066667 ; 1 / 15  
 I[44][16] = 0.066667 ; 1 / 15  
 I[18][11] = 0.062500 ; 1 / 16  
 I[18][16] = 0.062500 ; 1 / 16  
 I[22][7] = 0.062500 ; 1 / 16  
 I[23][15] = 0.062500 ; 1 / 16  
 I[28][7] = 0.062500 ; 1 / 16  
 I[28][18] = 0.062500 ; 1 / 16  
 I[44][12] = 0.062500 ; 1 / 16  
 I[44][28] = 0.062500 ; 1 / 16  
 I[11][7] = 0.058824 ; 1 / 17  
 I[16][7] = 0.058824 ; 1 / 17  
 I[18][12] = 0.058824 ; 1 / 17  
 I[43][7] = 0.058824 ; 1 / 17  
 I[45][24] = 0.058824 ; 1 / 17  
 I[45][26] = 0.058824 ; 1 / 17  
 I[18][15] = 0.055556 ; 1 / 18  
 I[23][7] = 0.055556 ; 1 / 18  
 I[44][23] = 0.055556 ; 1 / 18  
 I[45][5] = 0.055556 ; 1 / 18  
 I[23][18] = 0.052632 ; 1 / 19  
 I[45][28] = 0.052632 ; 1 / 19  
 I[45][39] = 0.052632 ; 1 / 19  
 I[45][40] = 0.052632 ; 1 / 19  
 I[45][41] = 0.052632 ; 1 / 19  
 I[45][42] = 0.052632 ; 1 / 19  
 I[45][12] = 0.050000 ; 1 / 20  
 I[22][9] = 0.047619 ; 1 / 21  
 I[13][9] = 0.041667 ; 1 / 24  
 I[43][9] = 0.041667 ; 1 / 24  
 I[15][9] = 0.040000 ; 1 / 25  
 I[23][9] = 0.040000 ; 1 / 25  
 JLL OTHER EDGES = 0.000000

Node[12] = 40 , I[12] = 0.500000  
 Node[13] = 39 , I[13] = 0.500000  
 Node[14] = 41 , I[14] = 0.500000  
 Node[15] = 42 , I[15] = 0.500000  
 Node[16] = 45 , I[16] = 0.470588  
 Node[17] = 9 , I[17] = 0.428571  
 Node[18] = 26 , I[18] = 0.428571  
 Node[19] = 5 , I[19] = 0.428571  
 Node[20] = 38 , I[20] = 0.428571  
 Node[21] = 16 , I[21] = 0.333333  
 Node[22] = 11 , I[22] = 0.333333  
 Node[23] = 24 , I[23] = 0.333333  
 Node[24] = 28 , I[24] = 0.333333  
 Node[25] = 3 , I[25] = 0.285714  
 Node[26] = 15 , I[26] = 0.272727  
 Node[27] = 10 , I[27] = 0.250000  
 Node[28] = 25 , I[28] = 0.250000  
 Node[29] = 6 , I[29] = 0.250000  
 Node[30] = 22 , I[30] = 0.200000  
 Node[31] = 23 , I[31] = 0.200000  
 Node[32] = 12 , I[32] = 0.166667  
 Node[33] = 4 , I[33] = 0.166667  
 Node[34] = 43 , I[34] = 0.142857  
 Node[35] = 13 , I[35] = 0.041667  
 Node[36] = 17 , I[36] = 0.000000  
 Node[37] = 21 , I[37] = 0.000000

Node[1] = 44 , I[1] = 0.909091  
 Node[2] = 18 , I[2] = 0.909091  
 Node[3] = 7 , I[3] = 0.615385  
 Node[4] = 2 , I[4] = 0.600000  
 Node[5] = 1 , I[5] = 0.600000  
 Node[6] = 33 , I[6] = 0.500000  
 Node[7] = 32 , I[7] = 0.500000  
 Node[8] = 34 , I[8] = 0.500000  
 Node[9] = 35 , I[9] = 0.500000  
 Node[10] = 36 , I[10] = 0.500000  
 Node[11] = 37 , I[11] = 0.500000



**VERSION 3  
COINCIDENTAL**

C[21][17] = 1.000000 ; 2 / 2	C[42][21] = 0.400000 ; 2 / 5
C[32][24] = 0.750000 ; 3 / 4	C[45][9] = 0.380952 ; 8 / 21
C[16][11] = 0.666667 ; 4 / 6	C[26][3] = 0.375000 ; 3 / 8
C[32][17] = 0.666667 ; 2 / 3	C[26][4] = 0.375000 ; 3 / 8
C[32][21] = 0.666667 ; 2 / 3	C[38][3] = 0.375000 ; 3 / 8
C[38][4] = 0.666667 ; 4 / 6	C[26][15] = 0.363636 ; 4 / 11
C[33][24] = 0.600000 ; 3 / 5	C[39][15] = 0.363636 ; 4 / 11
C[34][24] = 0.600000 ; 3 / 5	C[40][15] = 0.363636 ; 4 / 11
C[35][24] = 0.600000 ; 3 / 5	C[41][15] = 0.363636 ; 4 / 11
C[36][24] = 0.600000 ; 3 / 5	C[42][15] = 0.363636 ; 4 / 11
C[37][24] = 0.600000 ; 3 / 5	C[4][3] = 0.333333 ; 3 / 9
C[15][4] = 0.500000 ; 5 / 10	C[5][3] = 0.333333 ; 3 / 9
C[17][10] = 0.500000 ; 1 / 2	C[5][4] = 0.333333 ; 3 / 9
C[21][10] = 0.500000 ; 1 / 2	C[12][11] = 0.333333 ; 3 / 9
C[24][3] = 0.500000 ; 3 / 6	C[16][12] = 0.333333 ; 3 / 9
C[24][17] = 0.500000 ; 2 / 4	C[17][3] = 0.333333 ; 2 / 6
C[24][21] = 0.500000 ; 2 / 4	C[17][4] = 0.333333 ; 2 / 6
C[25][6] = 0.500000 ; 2 / 4	C[21][3] = 0.333333 ; 2 / 6
C[33][17] = 0.500000 ; 2 / 4	C[21][4] = 0.333333 ; 2 / 6
C[33][21] = 0.500000 ; 2 / 4	C[26][17] = 0.333333 ; 2 / 6
C[34][17] = 0.500000 ; 2 / 4	C[26][21] = 0.333333 ; 2 / 6
C[34][21] = 0.500000 ; 2 / 4	C[26][24] = 0.333333 ; 2 / 6
C[35][17] = 0.500000 ; 2 / 4	C[28][5] = 0.333333 ; 3 / 9
C[35][21] = 0.500000 ; 2 / 4	C[32][3] = 0.333333 ; 2 / 6
C[36][3] = 0.500000 ; 3 / 6	C[32][10] = 0.333333 ; 1 / 3
C[36][17] = 0.500000 ; 2 / 4	C[32][26] = 0.333333 ; 2 / 6
C[36][21] = 0.500000 ; 2 / 4	C[38][12] = 0.333333 ; 3 / 9
C[37][17] = 0.500000 ; 2 / 4	C[38][32] = 0.333333 ; 2 / 6
C[37][21] = 0.500000 ; 2 / 4	C[39][32] = 0.333333 ; 2 / 6
C[38][15] = 0.500000 ; 5 / 10	C[39][38] = 0.333333 ; 2 / 6
C[15][5] = 0.454545 ; 5 / 11	C[40][32] = 0.333333 ; 2 / 6
C[28][12] = 0.444444 ; 4 / 9	C[40][38] = 0.333333 ; 2 / 6
C[24][4] = 0.428571 ; 3 / 7	C[41][32] = 0.333333 ; 2 / 6
C[39][4] = 0.428571 ; 3 / 7	C[41][38] = 0.333333 ; 2 / 6
C[40][4] = 0.428571 ; 3 / 7	C[42][32] = 0.333333 ; 2 / 6
C[41][4] = 0.428571 ; 3 / 7	C[42][38] = 0.333333 ; 2 / 6
C[42][4] = 0.428571 ; 3 / 7	C[28][15] = 0.307692 ; 4 / 13
C[38][17] = 0.400000 ; 2 / 5	C[12][4] = 0.300000 ; 3 / 10
C[38][21] = 0.400000 ; 2 / 5	C[12][5] = 0.300000 ; 3 / 10
C[39][17] = 0.400000 ; 2 / 5	C[24][15] = 0.300000 ; 3 / 10
C[39][21] = 0.400000 ; 2 / 5	C[26][12] = 0.300000 ; 3 / 10
C[40][17] = 0.400000 ; 2 / 5	C[28][3] = 0.300000 ; 3 / 10
C[40][21] = 0.400000 ; 2 / 5	C[43][3] = 0.300000 ; 3 / 10
C[41][17] = 0.400000 ; 2 / 5	C[45][7] = 0.294118 ; 5 / 17
C[41][21] = 0.400000 ; 2 / 5	C[45][26] = 0.294118 ; 5 / 17
C[42][17] = 0.400000 ; 2 / 5	C[6][3] = 0.285714 ; 2 / 7
	C[24][5] = 0.285714 ; 2 / 7
	C[26][5] = 0.285714 ; 2 / 7
	C[28][9] = 0.285714 ; 6 / 21
	C[32][4] = 0.285714 ; 2 / 7
	C[32][5] = 0.285714 ; 2 / 7

C[33][3] = 0.285714 ; 2 / 7  
 C[33][26] = 0.285714 ; 2 / 7  
 C[34][3] = 0.285714 ; 2 / 7  
 C[34][26] = 0.285714 ; 2 / 7  
 C[35][1] = 0.285714 ; 2 / 7  
 C[35][3] = 0.285714 ; 2 / 7  
 C[35][26] = 0.285714 ; 2 / 7  
 C[36][1] = 0.285714 ; 2 / 7  
 C[36][26] = 0.285714 ; 2 / 7  
 C[37][3] = 0.285714 ; 2 / 7  
 C[37][26] = 0.285714 ; 2 / 7  
 C[38][7] = 0.285714 ; 4 / 14  
 C[38][33] = 0.285714 ; 2 / 7  
 C[38][34] = 0.285714 ; 2 / 7  
 C[38][35] = 0.285714 ; 2 / 7  
 C[38][36] = 0.285714 ; 2 / 7  
 C[38][37] = 0.285714 ; 2 / 7  
 C[39][33] = 0.285714 ; 2 / 7  
 C[39][34] = 0.285714 ; 2 / 7  
 C[39][35] = 0.285714 ; 2 / 7  
 C[39][36] = 0.285714 ; 2 / 7  
 C[39][37] = 0.285714 ; 2 / 7  
 C[40][33] = 0.285714 ; 2 / 7  
 C[40][34] = 0.285714 ; 2 / 7  
 C[40][35] = 0.285714 ; 2 / 7  
 C[40][36] = 0.285714 ; 2 / 7  
 C[40][37] = 0.285714 ; 2 / 7  
 C[41][33] = 0.285714 ; 2 / 7  
 C[41][34] = 0.285714 ; 2 / 7  
 C[41][35] = 0.285714 ; 2 / 7  
 C[41][36] = 0.285714 ; 2 / 7  
 C[41][37] = 0.285714 ; 2 / 7  
 C[42][33] = 0.285714 ; 2 / 7  
 C[42][34] = 0.285714 ; 2 / 7  
 C[42][35] = 0.285714 ; 2 / 7  
 C[42][36] = 0.285714 ; 2 / 7  
 C[42][37] = 0.285714 ; 2 / 7  
 C[43][6] = 0.285714 ; 2 / 7  
 C[45][15] = 0.285714 ; 6 / 21  
 C[15][7] = 0.277778 ; 5 / 18  
 C[45][3] = 0.277778 ; 5 / 18  
 C[7][3] = 0.266667 ; 4 / 15  
 C[7][4] = 0.266667 ; 4 / 15  
 C[26][7] = 0.266667 ; 4 / 15  
 C[23][4] = 0.250000 ; 3 / 12  
 C[25][5] = 0.250000 ; 2 / 8  
 C[28][23] = 0.250000 ; 3 / 12  
 C[28][24] = 0.250000 ; 2 / 8  
 C[32][28] = 0.250000 ; 2 / 8  
 C[33][4] = 0.250000 ; 2 / 8  
 C[33][5] = 0.250000 ; 2 / 8

C[33][10] = 0.250000 ; 1 / 4  
 C[33][32] = 0.250000 ; 1 / 4  
 C[34][4] = 0.250000 ; 2 / 8  
 C[34][5] = 0.250000 ; 2 / 8  
 C[34][10] = 0.250000 ; 1 / 4  
 C[34][32] = 0.250000 ; 1 / 4  
 C[35][4] = 0.250000 ; 2 / 8  
 C[35][5] = 0.250000 ; 2 / 8  
 C[35][10] = 0.250000 ; 1 / 4  
 C[35][32] = 0.250000 ; 1 / 4  
 C[36][4] = 0.250000 ; 2 / 8  
 C[36][5] = 0.250000 ; 2 / 8  
 C[36][10] = 0.250000 ; 1 / 4  
 C[36][32] = 0.250000 ; 1 / 4  
 C[37][4] = 0.250000 ; 2 / 8  
 C[37][5] = 0.250000 ; 2 / 8  
 C[37][10] = 0.250000 ; 1 / 4  
 C[37][32] = 0.250000 ; 1 / 4  
 C[38][1] = 0.250000 ; 2 / 8  
 C[43][25] = 0.250000 ; 2 / 8  
 C[43][32] = 0.250000 ; 2 / 8  
 C[9][5] = 0.238095 ; 5 / 21  
 C[26][9] = 0.238095 ; 5 / 21  
 C[4][1] = 0.222222 ; 2 / 9  
 C[5][1] = 0.222222 ; 2 / 9  
 C[12][1] = 0.222222 ; 2 / 9  
 C[24][12] = 0.222222 ; 2 / 9  
 C[26][1] = 0.222222 ; 2 / 9  
 C[28][1] = 0.222222 ; 2 / 9  
 C[33][28] = 0.222222 ; 2 / 9  
 C[34][28] = 0.222222 ; 2 / 9  
 C[35][28] = 0.222222 ; 2 / 9  
 C[36][28] = 0.222222 ; 2 / 9  
 C[37][28] = 0.222222 ; 2 / 9  
 C[39][3] = 0.222222 ; 2 / 9  
 C[40][3] = 0.222222 ; 2 / 9  
 C[41][3] = 0.222222 ; 2 / 9  
 C[42][3] = 0.222222 ; 2 / 9  
 C[43][24] = 0.222222 ; 2 / 9  
 C[43][33] = 0.222222 ; 2 / 9  
 C[43][34] = 0.222222 ; 2 / 9  
 C[43][35] = 0.222222 ; 2 / 9  
 C[43][36] = 0.222222 ; 2 / 9  
 C[43][37] = 0.222222 ; 2 / 9  
 C[45][5] = 0.222222 ; 4 / 18  
 C[15][12] = 0.214286 ; 3 / 14  
 C[18][3] = 0.214286 ; 3 / 14  
 C[24][7] = 0.214286 ; 3 / 14  
 C[43][15] = 0.214286 ; 3 / 14  
 C[45][4] = 0.210526 ; 4 / 19  
 C[45][28] = 0.210526 ; 4 / 19

C[10][1] = 0.200000 ; 1 / 5  
 C[17][15] = 0.200000 ; 2 / 10  
 C[21][15] = 0.200000 ; 2 / 10  
 C[22][6] = 0.200000 ; 1 / 5  
 C[28][4] = 0.200000 ; 2 / 10  
 C[28][11] = 0.200000 ; 2 / 10  
 C[28][16] = 0.200000 ; 2 / 10  
 C[32][6] = 0.200000 ; 1 / 5  
 C[32][15] = 0.200000 ; 2 / 10  
 C[34][33] = 0.200000 ; 1 / 5  
 C[35][33] = 0.200000 ; 1 / 5  
 C[35][34] = 0.200000 ; 1 / 5  
 C[36][33] = 0.200000 ; 1 / 5  
 C[36][34] = 0.200000 ; 1 / 5  
 C[36][35] = 0.200000 ; 1 / 5  
 C[37][33] = 0.200000 ; 1 / 5  
 C[37][34] = 0.200000 ; 1 / 5  
 C[37][35] = 0.200000 ; 1 / 5  
 C[37][36] = 0.200000 ; 1 / 5  
 C[39][7] = 0.200000 ; 3 / 15  
 C[39][12] = 0.200000 ; 2 / 10  
 C[40][7] = 0.200000 ; 3 / 15  
 C[40][12] = 0.200000 ; 2 / 10  
 C[41][7] = 0.200000 ; 3 / 15  
 C[41][12] = 0.200000 ; 2 / 10  
 C[42][7] = 0.200000 ; 3 / 15  
 C[42][12] = 0.200000 ; 2 / 10  
 C[45][43] = 0.200000 ; 4 / 20  
 C[9][7] = 0.190476 ; 4 / 21  
 C[7][5] = 0.187500 ; 3 / 16  
 C[28][7] = 0.187500 ; 3 / 16  
 C[9][3] = 0.181818 ; 4 / 22  
 C[12][3] = 0.181818 ; 2 / 11  
 C[15][3] = 0.181818 ; 2 / 11  
 C[15][6] = 0.181818 ; 2 / 11  
 C[18][6] = 0.181818 ; 2 / 11  
 C[25][15] = 0.181818 ; 2 / 11  
 C[33][15] = 0.181818 ; 2 / 11  
 C[34][15] = 0.181818 ; 2 / 11  
 C[35][15] = 0.181818 ; 2 / 11  
 C[36][15] = 0.181818 ; 2 / 11  
 C[37][15] = 0.181818 ; 2 / 11  
 C[38][23] = 0.181818 ; 2 / 11  
 C[43][26] = 0.181818 ; 2 / 11  
 C[45][23] = 0.181818 ; 4 / 22  
 C[45][6] = 0.176471 ; 3 / 17  
 C[45][18] = 0.176471 ; 3 / 17  
 C[45][24] = 0.176471 ; 3 / 17  
 C[45][32] = 0.176471 ; 3 / 17  
 C[10][3] = 0.166667 ; 1 / 6  
 C[10][4] = 0.166667 ; 1 / 6

C[11][10] = 0.166667 ; 1 / 6  
 C[16][10] = 0.166667 ; 1 / 6  
 C[17][1] = 0.166667 ; 1 / 6  
 C[21][1] = 0.166667 ; 1 / 6  
 C[23][5] = 0.166667 ; 2 / 12  
 C[23][7] = 0.166667 ; 3 / 18  
 C[23][12] = 0.166667 ; 2 / 12  
 C[24][6] = 0.166667 ; 1 / 6  
 C[25][18] = 0.166667 ; 2 / 12  
 C[25][22] = 0.166667 ; 1 / 6  
 C[26][23] = 0.166667 ; 2 / 12  
 C[28][13] = 0.166667 ; 2 / 12  
 C[32][18] = 0.166667 ; 2 / 12  
 C[32][22] = 0.166667 ; 1 / 6  
 C[32][25] = 0.166667 ; 1 / 6  
 C[33][6] = 0.166667 ; 1 / 6  
 C[34][6] = 0.166667 ; 1 / 6  
 C[35][6] = 0.166667 ; 1 / 6  
 C[36][2] = 0.166667 ; 1 / 6  
 C[36][6] = 0.166667 ; 1 / 6  
 C[37][6] = 0.166667 ; 1 / 6  
 C[38][24] = 0.166667 ; 1 / 6  
 C[40][39] = 0.166667 ; 1 / 6  
 C[41][39] = 0.166667 ; 1 / 6  
 C[41][40] = 0.166667 ; 1 / 6  
 C[42][39] = 0.166667 ; 1 / 6  
 C[42][40] = 0.166667 ; 1 / 6  
 C[42][41] = 0.166667 ; 1 / 6  
 C[43][13] = 0.166667 ; 2 / 12  
 C[45][22] = 0.166667 ; 3 / 18  
 C[45][25] = 0.166667 ; 3 / 18  
 C[45][33] = 0.166667 ; 3 / 18  
 C[45][34] = 0.166667 ; 3 / 18  
 C[45][35] = 0.166667 ; 3 / 18  
 C[45][36] = 0.166667 ; 3 / 18  
 C[45][37] = 0.166667 ; 3 / 18  
 C[15][9] = 0.160000 ; 4 / 25  
 C[45][1] = 0.157895 ; 3 / 19  
 C[17][7] = 0.153846 ; 2 / 13  
 C[21][7] = 0.153846 ; 2 / 13  
 C[33][18] = 0.153846 ; 2 / 13  
 C[34][18] = 0.153846 ; 2 / 13  
 C[35][18] = 0.153846 ; 2 / 13  
 C[36][18] = 0.153846 ; 2 / 13  
 C[37][18] = 0.153846 ; 2 / 13  
 C[44][7] = 0.153846 ; 2 / 13  
 C[9][6] = 0.150000 ; 3 / 20  
 C[45][12] = 0.150000 ; 3 / 20  
 C[12][10] = 0.142857 ; 1 / 7  
 C[17][5] = 0.142857 ; 1 / 7  
 C[17][11] = 0.142857 ; 1 / 7

C[17][16] = 0.142857 ; 1 / 7  
 C[18][7] = 0.142857 ; 2 / 14  
 C[21][5] = 0.142857 ; 1 / 7  
 C[21][11] = 0.142857 ; 1 / 7  
 C[21][16] = 0.142857 ; 1 / 7  
 C[24][1] = 0.142857 ; 1 / 7  
 C[24][9] = 0.142857 ; 3 / 21  
 C[24][22] = 0.142857 ; 1 / 7  
 C[25][9] = 0.142857 ; 3 / 21  
 C[25][24] = 0.142857 ; 1 / 7  
 C[28][10] = 0.142857 ; 1 / 7  
 C[32][1] = 0.142857 ; 1 / 7  
 C[32][7] = 0.142857 ; 2 / 14  
 C[33][22] = 0.142857 ; 1 / 7  
 C[33][25] = 0.142857 ; 1 / 7  
 C[34][22] = 0.142857 ; 1 / 7  
 C[34][25] = 0.142857 ; 1 / 7  
 C[35][22] = 0.142857 ; 1 / 7  
 C[35][25] = 0.142857 ; 1 / 7  
 C[36][22] = 0.142857 ; 1 / 7  
 C[36][25] = 0.142857 ; 1 / 7  
 C[37][22] = 0.142857 ; 1 / 7  
 C[37][25] = 0.142857 ; 1 / 7  
 C[38][2] = 0.142857 ; 1 / 7  
 C[38][26] = 0.142857 ; 1 / 7  
 C[39][24] = 0.142857 ; 1 / 7  
 C[40][24] = 0.142857 ; 1 / 7  
 C[41][24] = 0.142857 ; 1 / 7  
 C[42][24] = 0.142857 ; 1 / 7  
 C[43][18] = 0.142857 ; 2 / 14  
 C[44][3] = 0.142857 ; 2 / 14  
 C[9][1] = 0.136364 ; 3 / 22  
 C[38][9] = 0.136364 ; 3 / 22  
 C[7][1] = 0.133333 ; 2 / 15  
 C[26][18] = 0.133333 ; 2 / 15  
 C[33][7] = 0.133333 ; 2 / 15  
 C[34][7] = 0.133333 ; 2 / 15  
 C[35][7] = 0.133333 ; 2 / 15  
 C[36][7] = 0.133333 ; 2 / 15  
 C[37][7] = 0.133333 ; 2 / 15  
 C[9][4] = 0.130435 ; 3 / 23  
 C[3][1] = 0.125000 ; 1 / 8  
 C[4][2] = 0.125000 ; 1 / 8  
 C[5][2] = 0.125000 ; 1 / 8  
 C[6][5] = 0.125000 ; 1 / 8  
 C[12][2] = 0.125000 ; 1 / 8  
 C[12][7] = 0.125000 ; 2 / 16  
 C[17][12] = 0.125000 ; 1 / 8  
 C[21][12] = 0.125000 ; 1 / 8  
 C[23][15] = 0.125000 ; 2 / 16  
 C[26][2] = 0.125000 ; 1 / 8

C[26][6] = 0.125000 ; 1 / 8  
 C[28][2] = 0.125000 ; 1 / 8  
 C[28][6] = 0.125000 ; 1 / 8  
 C[28][17] = 0.125000 ; 1 / 8  
 C[28][21] = 0.125000 ; 1 / 8  
 C[32][11] = 0.125000 ; 1 / 8  
 C[32][16] = 0.125000 ; 1 / 8  
 C[33][1] = 0.125000 ; 1 / 8  
 C[34][1] = 0.125000 ; 1 / 8  
 C[37][1] = 0.125000 ; 1 / 8  
 C[38][5] = 0.125000 ; 1 / 8  
 C[39][26] = 0.125000 ; 1 / 8  
 C[40][26] = 0.125000 ; 1 / 8  
 C[41][26] = 0.125000 ; 1 / 8  
 C[42][26] = 0.125000 ; 1 / 8  
 C[43][17] = 0.125000 ; 1 / 8  
 C[43][21] = 0.125000 ; 1 / 8  
 C[23][9] = 0.120000 ; 3 / 25  
 C[43][7] = 0.117647 ; 2 / 17  
 C[45][17] = 0.117647 ; 2 / 17  
 C[45][21] = 0.117647 ; 2 / 17  
 C[45][44] = 0.117647 ; 2 / 17  
 C[11][1] = 0.111111 ; 1 / 9  
 C[13][2] = 0.111111 ; 1 / 9  
 C[13][6] = 0.111111 ; 1 / 9  
 C[16][1] = 0.111111 ; 1 / 9  
 C[18][15] = 0.111111 ; 2 / 18  
 C[22][3] = 0.111111 ; 1 / 9  
 C[22][5] = 0.111111 ; 1 / 9  
 C[24][11] = 0.111111 ; 1 / 9  
 C[24][16] = 0.111111 ; 1 / 9  
 C[26][22] = 0.111111 ; 1 / 9  
 C[26][25] = 0.111111 ; 1 / 9  
 C[28][22] = 0.111111 ; 1 / 9  
 C[28][25] = 0.111111 ; 1 / 9  
 C[28][26] = 0.111111 ; 1 / 9  
 C[32][12] = 0.111111 ; 1 / 9  
 C[33][11] = 0.111111 ; 1 / 9  
 C[33][16] = 0.111111 ; 1 / 9  
 C[34][11] = 0.111111 ; 1 / 9  
 C[34][16] = 0.111111 ; 1 / 9  
 C[35][11] = 0.111111 ; 1 / 9  
 C[35][16] = 0.111111 ; 1 / 9  
 C[36][11] = 0.111111 ; 1 / 9  
 C[36][16] = 0.111111 ; 1 / 9  
 C[37][11] = 0.111111 ; 1 / 9  
 C[37][16] = 0.111111 ; 1 / 9  
 C[38][28] = 0.111111 ; 1 / 9  
 C[39][1] = 0.111111 ; 1 / 9  
 C[40][1] = 0.111111 ; 1 / 9  
 C[41][1] = 0.111111 ; 1 / 9

C[42][1] = 0.111111 ; 1 / 9  
 C[43][22] = 0.111111 ; 1 / 9  
 C[44][15] = 0.111111 ; 2 / 18  
 C[45][2] = 0.111111 ; 2 / 18  
 C[45][38] = 0.111111 ; 2 / 18  
 C[45][39] = 0.105263 ; 2 / 19  
 C[45][40] = 0.105263 ; 2 / 19  
 C[45][41] = 0.105263 ; 2 / 19  
 C[45][42] = 0.105263 ; 2 / 19  
 C[15][10] = 0.100000 ; 1 / 10  
 C[22][13] = 0.100000 ; 1 / 10  
 C[23][17] = 0.100000 ; 1 / 10  
 C[23][21] = 0.100000 ; 1 / 10  
 C[25][13] = 0.100000 ; 1 / 10  
 C[33][12] = 0.100000 ; 1 / 10  
 C[34][12] = 0.100000 ; 1 / 10  
 C[35][12] = 0.100000 ; 1 / 10  
 C[36][12] = 0.100000 ; 1 / 10  
 C[37][12] = 0.100000 ; 1 / 10  
 C[38][11] = 0.100000 ; 1 / 10  
 C[38][16] = 0.100000 ; 1 / 10  
 C[39][11] = 0.100000 ; 1 / 10  
 C[39][16] = 0.100000 ; 1 / 10  
 C[39][28] = 0.100000 ; 1 / 10  
 C[40][11] = 0.100000 ; 1 / 10  
 C[40][16] = 0.100000 ; 1 / 10  
 C[40][28] = 0.100000 ; 1 / 10  
 C[41][11] = 0.100000 ; 1 / 10  
 C[41][16] = 0.100000 ; 1 / 10  
 C[41][28] = 0.100000 ; 1 / 10  
 C[42][11] = 0.100000 ; 1 / 10  
 C[42][16] = 0.100000 ; 1 / 10  
 C[42][28] = 0.100000 ; 1 / 10  
 C[9][2] = 0.095238 ; 2 / 21  
 C[22][9] = 0.095238 ; 2 / 21  
 C[32][9] = 0.095238 ; 2 / 21  
 C[45][11] = 0.095238 ; 2 / 21  
 C[45][16] = 0.095238 ; 2 / 21  
 C[11][3] = 0.090909 ; 1 / 11  
 C[11][4] = 0.090909 ; 1 / 11  
 C[11][5] = 0.090909 ; 1 / 11  
 C[13][1] = 0.090909 ; 1 / 11  
 C[16][3] = 0.090909 ; 1 / 11  
 C[16][4] = 0.090909 ; 1 / 11  
 C[16][5] = 0.090909 ; 1 / 11  
 C[23][1] = 0.090909 ; 1 / 11  
 C[26][11] = 0.090909 ; 1 / 11  
 C[26][16] = 0.090909 ; 1 / 11  
 C[32][23] = 0.090909 ; 1 / 11  
 C[33][9] = 0.090909 ; 2 / 22  
 C[34][9] = 0.090909 ; 2 / 22

C[35][9] = 0.090909 ; 2 / 22  
 C[36][9] = 0.090909 ; 2 / 22  
 C[37][9] = 0.090909 ; 2 / 22  
 C[43][38] = 0.090909 ; 1 / 11  
 C[44][6] = 0.090909 ; 1 / 11  
 C[44][17] = 0.090909 ; 1 / 11  
 C[44][21] = 0.090909 ; 1 / 11  
 C[12][9] = 0.086957 ; 2 / 23  
 C[39][9] = 0.086957 ; 2 / 23  
 C[40][9] = 0.086957 ; 2 / 23  
 C[41][9] = 0.086957 ; 2 / 23  
 C[42][9] = 0.086957 ; 2 / 23  
 C[11][9] = 0.083333 ; 2 / 24  
 C[13][9] = 0.083333 ; 2 / 24  
 C[13][11] = 0.083333 ; 1 / 12  
 C[16][9] = 0.083333 ; 2 / 24  
 C[16][13] = 0.083333 ; 1 / 12  
 C[18][17] = 0.083333 ; 1 / 12  
 C[21][18] = 0.083333 ; 1 / 12  
 C[33][23] = 0.083333 ; 1 / 12  
 C[34][23] = 0.083333 ; 1 / 12  
 C[35][23] = 0.083333 ; 1 / 12  
 C[36][23] = 0.083333 ; 1 / 12  
 C[37][23] = 0.083333 ; 1 / 12  
 C[39][23] = 0.083333 ; 1 / 12  
 C[40][23] = 0.083333 ; 1 / 12  
 C[41][23] = 0.083333 ; 1 / 12  
 C[42][23] = 0.083333 ; 1 / 12  
 C[43][5] = 0.083333 ; 1 / 12  
 C[43][9] = 0.083333 ; 2 / 24  
 C[43][28] = 0.083333 ; 1 / 12  
 C[44][25] = 0.083333 ; 1 / 12  
 C[44][32] = 0.083333 ; 1 / 12  
 C[10][7] = 0.076923 ; 1 / 13  
 C[13][12] = 0.076923 ; 1 / 13  
 C[15][1] = 0.076923 ; 1 / 13  
 C[23][3] = 0.076923 ; 1 / 13  
 C[23][11] = 0.076923 ; 1 / 13  
 C[23][16] = 0.076923 ; 1 / 13  
 C[24][18] = 0.076923 ; 1 / 13  
 C[44][33] = 0.076923 ; 1 / 13  
 C[44][34] = 0.076923 ; 1 / 13  
 C[44][35] = 0.076923 ; 1 / 13  
 C[44][36] = 0.076923 ; 1 / 13  
 C[44][37] = 0.076923 ; 1 / 13  
 C[7][2] = 0.071429 ; 1 / 14  
 C[7][6] = 0.071429 ; 1 / 14  
 C[44][39] = 0.071429 ; 1 / 14  
 C[44][40] = 0.071429 ; 1 / 14  
 C[44][41] = 0.071429 ; 1 / 14  
 C[44][42] = 0.071429 ; 1 / 14

C[44][43] = 0.071429 ; 1 / 14  
 C[23][13] = 0.066667 ; 1 / 15  
 C[25][7] = 0.066667 ; 1 / 15  
 C[39][18] = 0.066667 ; 1 / 15  
 C[40][18] = 0.066667 ; 1 / 15  
 C[41][18] = 0.066667 ; 1 / 15  
 C[42][18] = 0.066667 ; 1 / 15  
 C[44][4] = 0.066667 ; 1 / 15  
 C[44][26] = 0.066667 ; 1 / 15  
 C[18][4] = 0.062500 ; 1 / 16  
 C[18][5] = 0.062500 ; 1 / 16  
 C[28][18] = 0.062500 ; 1 / 16  
 C[44][13] = 0.062500 ; 1 / 16  
 C[11][7] = 0.058824 ; 1 / 17  
 C[16][7] = 0.058824 ; 1 / 17  
 C[18][13] = 0.058824 ; 1 / 17  
 C[45][10] = 0.058824 ; 1 / 17  
 C[13][7] = 0.052632 ; 1 / 19  
 C[10][9] = 0.050000 ; 1 / 20  
 C[17][9] = 0.047619 ; 1 / 21  
 C[18][9] = 0.047619 ; 1 / 21  
 C[21][9] = 0.047619 ; 1 / 21  
 C[44][9] = 0.047619 ; 1 / 21  
 C[45][13] = 0.043478 ; 1 / 23  
 JLL OTHER EDGES = 0.000000

Node[26] = 45 , C[26] = 0.380952  
 Node[27] = 9 , C[27] = 0.380952  
 Node[28] = 26 , C[28] = 0.375000  
 Node[29] = 43 , C[29] = 0.300000  
 Node[30] = 7 , C[30] = 0.294118  
 Node[31] = 1 , C[31] = 0.285714  
 Node[32] = 23 , C[32] = 0.250000  
 Node[33] = 18 , C[33] = 0.214286  
 Node[34] = 22 , C[34] = 0.200000  
 Node[35] = 13 , C[35] = 0.166667  
 Node[36] = 2 , C[36] = 0.166667  
 Node[37] = 44 , C[37] = 0.153846

Node[1] = 21 , C[1] = 1.000000  
 Node[2] = 17 , C[2] = 1.000000  
 Node[3] = 32 , C[3] = 0.750000  
 Node[4] = 24 , C[4] = 0.750000  
 Node[5] = 16 , C[5] = 0.666667  
 Node[6] = 11 , C[6] = 0.666667  
 Node[7] = 38 , C[7] = 0.666667  
 Node[8] = 4 , C[8] = 0.666667  
 Node[9] = 33 , C[9] = 0.600000  
 Node[10] = 34 , C[10] = 0.600000  
 Node[11] = 35 , C[11] = 0.600000  
 Node[12] = 36 , C[12] = 0.600000  
 Node[13] = 37 , C[13] = 0.600000  
 Node[14] = 15 , C[14] = 0.500000  
 Node[15] = 10 , C[15] = 0.500000  
 Node[16] = 3 , C[16] = 0.500000  
 Node[17] = 25 , C[17] = 0.500000  
 Node[18] = 6 , C[18] = 0.500000  
 Node[19] = 5 , C[19] = 0.454545  
 Node[20] = 28 , C[20] = 0.444444  
 Node[21] = 12 , C[21] = 0.444444  
 Node[22] = 39 , C[22] = 0.428571  
 Node[23] = 40 , C[23] = 0.428571  
 Node[24] = 41 , C[24] = 0.428571  
 Node[25] = 42 , C[25] = 0.428571

**VERSION 3**  
**COMPOSITE**

J[16][11] = 1.000000 ; 6 / 6	J[37][34] = 0.600000 ; 3 / 5
J[21][17] = 1.000000 ; 2 / 2	J[37][35] = 0.600000 ; 3 / 5
J[44][18] = 0.909091 ; 10 / 11	J[37][36] = 0.600000 ; 3 / 5
J[38][4] = 0.833333 ; 5 / 6	J[45][44] = 0.588235 ; 10 / 17
J[44][7] = 0.769231 ; 10 / 13	J[9][7] = 0.571429 ; 12 / 21
J[45][7] = 0.764706 ; 13 / 17	J[38][26] = 0.571429 ; 4 / 7
J[45][9] = 0.761905 ; 16 / 21	J[39][4] = 0.571429 ; 4 / 7
J[25][6] = 0.750000 ; 3 / 4	J[40][4] = 0.571429 ; 4 / 7
J[32][24] = 0.750000 ; 3 / 4	J[41][4] = 0.571429 ; 4 / 7
J[33][32] = 0.750000 ; 3 / 4	J[42][4] = 0.571429 ; 4 / 7
J[34][32] = 0.750000 ; 3 / 4	J[28][12] = 0.555556 ; 5 / 9
J[35][32] = 0.750000 ; 3 / 4	J[17][10] = 0.500000 ; 1 / 2
J[36][32] = 0.750000 ; 3 / 4	J[21][10] = 0.500000 ; 1 / 2
J[37][32] = 0.750000 ; 3 / 4	J[24][17] = 0.500000 ; 2 / 4
J[18][7] = 0.714286 ; 10 / 14	J[24][21] = 0.500000 ; 2 / 4
J[26][5] = 0.714286 ; 5 / 7	J[26][3] = 0.500000 ; 4 / 8
J[24][3] = 0.666667 ; 4 / 6	J[26][4] = 0.500000 ; 4 / 8
J[26][24] = 0.666667 ; 4 / 6	J[32][3] = 0.500000 ; 3 / 6
J[32][17] = 0.666667 ; 2 / 3	J[32][26] = 0.500000 ; 3 / 6
J[32][21] = 0.666667 ; 2 / 3	J[33][17] = 0.500000 ; 2 / 4
J[36][3] = 0.666667 ; 4 / 6	J[33][21] = 0.500000 ; 2 / 4
J[39][38] = 0.666667 ; 4 / 6	J[34][17] = 0.500000 ; 2 / 4
J[40][38] = 0.666667 ; 4 / 6	J[34][21] = 0.500000 ; 2 / 4
J[40][39] = 0.666667 ; 4 / 6	J[35][17] = 0.500000 ; 2 / 4
J[41][38] = 0.666667 ; 4 / 6	J[35][21] = 0.500000 ; 2 / 4
J[41][39] = 0.666667 ; 4 / 6	J[36][17] = 0.500000 ; 2 / 4
J[41][40] = 0.666667 ; 4 / 6	J[36][21] = 0.500000 ; 2 / 4
J[42][38] = 0.666667 ; 4 / 6	J[37][17] = 0.500000 ; 2 / 4
J[42][39] = 0.666667 ; 4 / 6	J[37][21] = 0.500000 ; 2 / 4
J[42][40] = 0.666667 ; 4 / 6	J[38][15] = 0.500000 ; 5 / 10
J[42][41] = 0.666667 ; 4 / 6	J[38][24] = 0.500000 ; 3 / 6
J[45][18] = 0.647059 ; 11 / 17	J[18][9] = 0.476190 ; 10 / 21
J[2][1] = 0.600000 ; 3 / 5	J[15][3] = 0.454545 ; 5 / 11
J[15][4] = 0.600000 ; 6 / 10	J[15][5] = 0.454545 ; 5 / 11
J[33][24] = 0.600000 ; 3 / 5	J[26][15] = 0.454545 ; 5 / 11
J[34][24] = 0.600000 ; 3 / 5	J[12][11] = 0.444444 ; 4 / 9
J[34][33] = 0.600000 ; 3 / 5	J[16][12] = 0.444444 ; 4 / 9
J[35][24] = 0.600000 ; 3 / 5	J[28][5] = 0.444444 ; 4 / 9
J[35][33] = 0.600000 ; 3 / 5	J[28][26] = 0.444444 ; 4 / 9
J[35][34] = 0.600000 ; 3 / 5	J[24][4] = 0.428571 ; 3 / 7
J[36][24] = 0.600000 ; 3 / 5	J[24][5] = 0.428571 ; 3 / 7
J[36][33] = 0.600000 ; 3 / 5	J[33][3] = 0.428571 ; 3 / 7
J[36][34] = 0.600000 ; 3 / 5	J[33][26] = 0.428571 ; 3 / 7
J[36][35] = 0.600000 ; 3 / 5	J[34][3] = 0.428571 ; 3 / 7
J[37][24] = 0.600000 ; 3 / 5	J[34][26] = 0.428571 ; 3 / 7
J[37][33] = 0.600000 ; 3 / 5	J[35][3] = 0.428571 ; 3 / 7
	J[35][26] = 0.428571 ; 3 / 7
	J[36][26] = 0.428571 ; 3 / 7
	J[37][3] = 0.428571 ; 3 / 7
	J[37][26] = 0.428571 ; 3 / 7
	J[43][6] = 0.428571 ; 3 / 7

J[44][9] = 0.428571 ; 9 / 21  
 J[22][6] = 0.400000 ; 2 / 5  
 J[24][15] = 0.400000 ; 4 / 10  
 J[38][17] = 0.400000 ; 2 / 5  
 J[38][21] = 0.400000 ; 2 / 5  
 J[39][17] = 0.400000 ; 2 / 5  
 J[39][21] = 0.400000 ; 2 / 5  
 J[40][17] = 0.400000 ; 2 / 5  
 J[40][21] = 0.400000 ; 2 / 5  
 J[41][17] = 0.400000 ; 2 / 5  
 J[41][21] = 0.400000 ; 2 / 5  
 J[42][17] = 0.400000 ; 2 / 5  
 J[42][21] = 0.400000 ; 2 / 5  
 J[3][1] = 0.375000 ; 3 / 8  
 J[28][24] = 0.375000 ; 3 / 8  
 J[38][3] = 0.375000 ; 3 / 8  
 J[38][5] = 0.375000 ; 3 / 8  
 J[39][26] = 0.375000 ; 3 / 8  
 J[40][26] = 0.375000 ; 3 / 8  
 J[41][26] = 0.375000 ; 3 / 8  
 J[42][26] = 0.375000 ; 3 / 8  
 J[43][25] = 0.375000 ; 3 / 8  
 J[39][15] = 0.363636 ; 4 / 11  
 J[40][15] = 0.363636 ; 4 / 11  
 J[41][15] = 0.363636 ; 4 / 11  
 J[42][15] = 0.363636 ; 4 / 11  
 J[45][26] = 0.352941 ; 6 / 17  
 J[4][3] = 0.333333 ; 3 / 9  
 J[5][3] = 0.333333 ; 3 / 9  
 J[5][4] = 0.333333 ; 3 / 9  
 J[12][1] = 0.333333 ; 3 / 9  
 J[17][3] = 0.333333 ; 2 / 6  
 J[17][4] = 0.333333 ; 2 / 6  
 J[21][3] = 0.333333 ; 2 / 6  
 J[21][4] = 0.333333 ; 2 / 6  
 J[23][12] = 0.333333 ; 4 / 12  
 J[25][22] = 0.333333 ; 2 / 6  
 J[26][17] = 0.333333 ; 2 / 6  
 J[26][21] = 0.333333 ; 2 / 6  
 J[28][1] = 0.333333 ; 3 / 9  
 J[28][23] = 0.333333 ; 4 / 12  
 J[32][10] = 0.333333 ; 1 / 3  
 J[38][12] = 0.333333 ; 3 / 9  
 J[38][28] = 0.333333 ; 3 / 9  
 J[38][32] = 0.333333 ; 2 / 6  
 J[39][32] = 0.333333 ; 2 / 6  
 J[40][32] = 0.333333 ; 2 / 6  
 J[41][32] = 0.333333 ; 2 / 6  
 J[42][32] = 0.333333 ; 2 / 6  
 J[28][15] = 0.307692 ; 4 / 13  
 J[12][4] = 0.300000 ; 3 / 10

J[12][5] = 0.300000 ; 3 / 10  
 J[26][12] = 0.300000 ; 3 / 10  
 J[28][3] = 0.300000 ; 3 / 10  
 J[28][4] = 0.300000 ; 3 / 10  
 J[28][11] = 0.300000 ; 3 / 10  
 J[28][16] = 0.300000 ; 3 / 10  
 J[32][15] = 0.300000 ; 3 / 10  
 J[43][3] = 0.300000 ; 3 / 10  
 J[3][2] = 0.285714 ; 2 / 7  
 J[6][3] = 0.285714 ; 2 / 7  
 J[24][1] = 0.285714 ; 2 / 7  
 J[28][9] = 0.285714 ; 6 / 21  
 J[32][4] = 0.285714 ; 2 / 7  
 J[32][5] = 0.285714 ; 2 / 7  
 J[35][1] = 0.285714 ; 2 / 7  
 J[36][1] = 0.285714 ; 2 / 7  
 J[38][7] = 0.285714 ; 4 / 14  
 J[38][33] = 0.285714 ; 2 / 7  
 J[38][34] = 0.285714 ; 2 / 7  
 J[38][35] = 0.285714 ; 2 / 7  
 J[38][36] = 0.285714 ; 2 / 7  
 J[38][37] = 0.285714 ; 2 / 7  
 J[39][24] = 0.285714 ; 2 / 7  
 J[39][33] = 0.285714 ; 2 / 7  
 J[39][34] = 0.285714 ; 2 / 7  
 J[39][35] = 0.285714 ; 2 / 7  
 J[39][36] = 0.285714 ; 2 / 7  
 J[39][37] = 0.285714 ; 2 / 7  
 J[40][24] = 0.285714 ; 2 / 7  
 J[40][33] = 0.285714 ; 2 / 7  
 J[40][34] = 0.285714 ; 2 / 7  
 J[40][35] = 0.285714 ; 2 / 7  
 J[40][36] = 0.285714 ; 2 / 7  
 J[40][37] = 0.285714 ; 2 / 7  
 J[41][24] = 0.285714 ; 2 / 7  
 J[41][33] = 0.285714 ; 2 / 7  
 J[41][34] = 0.285714 ; 2 / 7  
 J[41][35] = 0.285714 ; 2 / 7  
 J[41][36] = 0.285714 ; 2 / 7  
 J[41][37] = 0.285714 ; 2 / 7  
 J[42][24] = 0.285714 ; 2 / 7  
 J[42][33] = 0.285714 ; 2 / 7  
 J[42][34] = 0.285714 ; 2 / 7  
 J[42][35] = 0.285714 ; 2 / 7  
 J[42][36] = 0.285714 ; 2 / 7  
 J[42][37] = 0.285714 ; 2 / 7  
 J[43][18] = 0.285714 ; 4 / 14  
 J[45][15] = 0.285714 ; 6 / 21  
 J[15][7] = 0.277778 ; 5 / 18  
 J[45][3] = 0.277778 ; 5 / 18  
 J[45][5] = 0.277778 ; 5 / 18



J[18][6] = 0.272727 ; 3 / 11  
 J[23][1] = 0.272727 ; 3 / 11  
 J[25][15] = 0.272727 ; 3 / 11  
 J[33][15] = 0.272727 ; 3 / 11  
 J[34][15] = 0.272727 ; 3 / 11  
 J[35][15] = 0.272727 ; 3 / 11  
 J[36][15] = 0.272727 ; 3 / 11  
 J[37][15] = 0.272727 ; 3 / 11  
 J[38][23] = 0.272727 ; 3 / 11  
 J[7][3] = 0.266667 ; 4 / 15  
 J[7][4] = 0.266667 ; 4 / 15  
 J[26][7] = 0.266667 ; 4 / 15  
 J[45][4] = 0.263158 ; 5 / 19  
 J[45][28] = 0.263158 ; 5 / 19  
 J[12][2] = 0.250000 ; 2 / 8  
 J[12][7] = 0.250000 ; 4 / 16  
 J[23][4] = 0.250000 ; 3 / 12  
 J[23][5] = 0.250000 ; 3 / 12  
 J[24][10] = 0.250000 ; 1 / 4  
 J[25][3] = 0.250000 ; 2 / 8  
 J[25][5] = 0.250000 ; 2 / 8  
 J[25][18] = 0.250000 ; 3 / 12  
 J[26][23] = 0.250000 ; 3 / 12  
 J[28][2] = 0.250000 ; 2 / 8  
 J[28][6] = 0.250000 ; 2 / 8  
 J[28][7] = 0.250000 ; 4 / 16  
 J[32][28] = 0.250000 ; 2 / 8  
 J[33][4] = 0.250000 ; 2 / 8  
 J[33][5] = 0.250000 ; 2 / 8  
 J[33][10] = 0.250000 ; 1 / 4  
 J[34][4] = 0.250000 ; 2 / 8  
 J[34][5] = 0.250000 ; 2 / 8  
 J[34][10] = 0.250000 ; 1 / 4  
 J[35][4] = 0.250000 ; 2 / 8  
 J[35][5] = 0.250000 ; 2 / 8  
 J[35][10] = 0.250000 ; 1 / 4  
 J[36][4] = 0.250000 ; 2 / 8  
 J[36][5] = 0.250000 ; 2 / 8  
 J[36][10] = 0.250000 ; 1 / 4  
 J[37][4] = 0.250000 ; 2 / 8  
 J[37][5] = 0.250000 ; 2 / 8  
 J[37][10] = 0.250000 ; 1 / 4  
 J[38][1] = 0.250000 ; 2 / 8  
 J[43][32] = 0.250000 ; 2 / 8  
 J[9][5] = 0.238095 ; 5 / 21  
 J[26][9] = 0.238095 ; 5 / 21  
 J[45][24] = 0.235294 ; 4 / 17  
 J[4][1] = 0.222222 ; 2 / 9  
 J[5][1] = 0.222222 ; 2 / 9  
 J[11][1] = 0.222222 ; 2 / 9  
 J[16][1] = 0.222222 ; 2 / 9

J[23][7] = 0.222222 ; 4 / 18  
 J[24][12] = 0.222222 ; 2 / 9  
 J[26][1] = 0.222222 ; 2 / 9  
 J[28][22] = 0.222222 ; 2 / 9  
 J[28][25] = 0.222222 ; 2 / 9  
 J[33][28] = 0.222222 ; 2 / 9  
 J[34][28] = 0.222222 ; 2 / 9  
 J[35][28] = 0.222222 ; 2 / 9  
 J[36][28] = 0.222222 ; 2 / 9  
 J[37][28] = 0.222222 ; 2 / 9  
 J[39][3] = 0.222222 ; 2 / 9  
 J[39][5] = 0.222222 ; 2 / 9  
 J[40][3] = 0.222222 ; 2 / 9  
 J[40][5] = 0.222222 ; 2 / 9  
 J[41][3] = 0.222222 ; 2 / 9  
 J[41][5] = 0.222222 ; 2 / 9  
 J[42][3] = 0.222222 ; 2 / 9  
 J[42][5] = 0.222222 ; 2 / 9  
 J[43][22] = 0.222222 ; 2 / 9  
 J[43][24] = 0.222222 ; 2 / 9  
 J[43][33] = 0.222222 ; 2 / 9  
 J[43][34] = 0.222222 ; 2 / 9  
 J[43][35] = 0.222222 ; 2 / 9  
 J[43][36] = 0.222222 ; 2 / 9  
 J[43][37] = 0.222222 ; 2 / 9  
 J[45][38] = 0.222222 ; 4 / 18  
 J[15][12] = 0.214286 ; 3 / 14  
 J[18][3] = 0.214286 ; 3 / 14  
 J[24][7] = 0.214286 ; 3 / 14  
 J[43][15] = 0.214286 ; 3 / 14  
 J[44][43] = 0.214286 ; 3 / 14  
 J[6][2] = 0.200000 ; 1 / 5  
 J[7][1] = 0.200000 ; 3 / 15  
 J[10][1] = 0.200000 ; 1 / 5  
 J[15][9] = 0.200000 ; 5 / 25  
 J[17][15] = 0.200000 ; 2 / 10  
 J[21][15] = 0.200000 ; 2 / 10  
 J[23][2] = 0.200000 ; 2 / 10  
 J[32][6] = 0.200000 ; 1 / 5  
 J[38][10] = 0.200000 ; 1 / 5  
 J[39][7] = 0.200000 ; 3 / 15  
 J[39][10] = 0.200000 ; 1 / 5  
 J[39][12] = 0.200000 ; 2 / 10  
 J[39][28] = 0.200000 ; 2 / 10  
 J[40][7] = 0.200000 ; 3 / 15  
 J[40][10] = 0.200000 ; 1 / 5  
 J[40][12] = 0.200000 ; 2 / 10  
 J[40][28] = 0.200000 ; 2 / 10  
 J[41][7] = 0.200000 ; 3 / 15  
 J[41][10] = 0.200000 ; 1 / 5  
 J[41][12] = 0.200000 ; 2 / 10

J[41][28] = 0.200000 ; 2 / 10  
 J[42][7] = 0.200000 ; 3 / 15  
 J[42][10] = 0.200000 ; 1 / 5  
 J[42][12] = 0.200000 ; 2 / 10  
 J[42][28] = 0.200000 ; 2 / 10  
 J[45][12] = 0.200000 ; 4 / 20  
 J[45][43] = 0.200000 ; 4 / 20  
 J[7][5] = 0.187500 ; 3 / 16  
 J[23][15] = 0.187500 ; 3 / 16  
 J[9][3] = 0.181818 ; 4 / 22  
 J[12][3] = 0.181818 ; 2 / 11  
 J[15][6] = 0.181818 ; 2 / 11  
 J[24][23] = 0.181818 ; 2 / 11  
 J[43][26] = 0.181818 ; 2 / 11  
 J[44][6] = 0.181818 ; 2 / 11  
 J[45][23] = 0.181818 ; 4 / 22  
 J[43][7] = 0.176471 ; 3 / 17  
 J[45][6] = 0.176471 ; 3 / 17  
 J[45][32] = 0.176471 ; 3 / 17  
 J[12][9] = 0.173913 ; 4 / 23  
 J[10][3] = 0.166667 ; 1 / 6  
 J[10][4] = 0.166667 ; 1 / 6  
 J[10][5] = 0.166667 ; 1 / 6  
 J[11][10] = 0.166667 ; 1 / 6  
 J[16][10] = 0.166667 ; 1 / 6  
 J[17][1] = 0.166667 ; 1 / 6  
 J[18][15] = 0.166667 ; 3 / 18  
 J[21][1] = 0.166667 ; 1 / 6  
 J[22][2] = 0.166667 ; 1 / 6  
 J[24][2] = 0.166667 ; 1 / 6  
 J[24][6] = 0.166667 ; 1 / 6  
 J[25][2] = 0.166667 ; 1 / 6  
 J[26][10] = 0.166667 ; 1 / 6  
 J[28][13] = 0.166667 ; 2 / 12  
 J[32][18] = 0.166667 ; 2 / 12  
 J[32][22] = 0.166667 ; 1 / 6  
 J[32][25] = 0.166667 ; 1 / 6  
 J[33][6] = 0.166667 ; 1 / 6  
 J[34][6] = 0.166667 ; 1 / 6  
 J[35][6] = 0.166667 ; 1 / 6  
 J[36][2] = 0.166667 ; 1 / 6  
 J[36][6] = 0.166667 ; 1 / 6  
 J[37][6] = 0.166667 ; 1 / 6  
 J[39][23] = 0.166667 ; 2 / 12  
 J[40][23] = 0.166667 ; 2 / 12  
 J[41][23] = 0.166667 ; 2 / 12  
 J[42][23] = 0.166667 ; 2 / 12  
 J[43][13] = 0.166667 ; 2 / 12  
 J[43][28] = 0.166667 ; 2 / 12  
 J[44][25] = 0.166667 ; 2 / 12  
 J[45][22] = 0.166667 ; 3 / 18

J[45][25] = 0.166667 ; 3 / 18  
 J[45][33] = 0.166667 ; 3 / 18  
 J[45][34] = 0.166667 ; 3 / 18  
 J[45][35] = 0.166667 ; 3 / 18  
 J[45][36] = 0.166667 ; 3 / 18  
 J[45][37] = 0.166667 ; 3 / 18  
 J[23][9] = 0.160000 ; 4 / 25  
 J[45][1] = 0.157895 ; 3 / 19  
 J[45][39] = 0.157895 ; 3 / 19  
 J[45][40] = 0.157895 ; 3 / 19  
 J[45][41] = 0.157895 ; 3 / 19  
 J[45][42] = 0.157895 ; 3 / 19  
 J[15][1] = 0.153846 ; 2 / 13  
 J[17][7] = 0.153846 ; 2 / 13  
 J[21][7] = 0.153846 ; 2 / 13  
 J[22][18] = 0.153846 ; 2 / 13  
 J[23][3] = 0.153846 ; 2 / 13  
 J[23][11] = 0.153846 ; 2 / 13  
 J[23][16] = 0.153846 ; 2 / 13  
 J[24][18] = 0.153846 ; 2 / 13  
 J[33][18] = 0.153846 ; 2 / 13  
 J[34][18] = 0.153846 ; 2 / 13  
 J[35][18] = 0.153846 ; 2 / 13  
 J[36][18] = 0.153846 ; 2 / 13  
 J[37][18] = 0.153846 ; 2 / 13  
 J[9][6] = 0.150000 ; 3 / 20  
 J[6][1] = 0.142857 ; 1 / 7  
 J[7][2] = 0.142857 ; 2 / 14  
 J[7][6] = 0.142857 ; 2 / 14  
 J[12][10] = 0.142857 ; 1 / 7  
 J[17][5] = 0.142857 ; 1 / 7  
 J[17][11] = 0.142857 ; 1 / 7  
 J[17][16] = 0.142857 ; 1 / 7  
 J[21][5] = 0.142857 ; 1 / 7  
 J[21][11] = 0.142857 ; 1 / 7  
 J[21][16] = 0.142857 ; 1 / 7  
 J[22][9] = 0.142857 ; 3 / 21  
 J[24][9] = 0.142857 ; 3 / 21  
 J[24][22] = 0.142857 ; 1 / 7  
 J[25][9] = 0.142857 ; 3 / 21  
 J[25][24] = 0.142857 ; 1 / 7  
 J[28][10] = 0.142857 ; 1 / 7  
 J[32][1] = 0.142857 ; 1 / 7  
 J[32][7] = 0.142857 ; 2 / 14  
 J[33][22] = 0.142857 ; 1 / 7  
 J[33][25] = 0.142857 ; 1 / 7  
 J[34][22] = 0.142857 ; 1 / 7  
 J[34][25] = 0.142857 ; 1 / 7  
 J[35][22] = 0.142857 ; 1 / 7  
 J[35][25] = 0.142857 ; 1 / 7  
 J[36][22] = 0.142857 ; 1 / 7

J[36][25] = 0.142857 ; 1 / 7  
 J[37][22] = 0.142857 ; 1 / 7  
 J[37][25] = 0.142857 ; 1 / 7  
 J[38][2] = 0.142857 ; 1 / 7  
 J[44][3] = 0.142857 ; 2 / 14  
 J[9][1] = 0.136364 ; 3 / 22  
 J[38][9] = 0.136364 ; 3 / 22  
 J[25][7] = 0.133333 ; 2 / 15  
 J[26][18] = 0.133333 ; 2 / 15  
 J[33][7] = 0.133333 ; 2 / 15  
 J[34][7] = 0.133333 ; 2 / 15  
 J[35][7] = 0.133333 ; 2 / 15  
 J[36][7] = 0.133333 ; 2 / 15  
 J[37][7] = 0.133333 ; 2 / 15  
 J[9][4] = 0.130435 ; 3 / 23  
 J[4][2] = 0.125000 ; 1 / 8  
 J[5][2] = 0.125000 ; 1 / 8  
 J[6][5] = 0.125000 ; 1 / 8  
 J[11][2] = 0.125000 ; 1 / 8  
 J[11][6] = 0.125000 ; 1 / 8  
 J[13][9] = 0.125000 ; 3 / 24  
 J[16][2] = 0.125000 ; 1 / 8  
 J[16][6] = 0.125000 ; 1 / 8  
 J[17][12] = 0.125000 ; 1 / 8  
 J[21][12] = 0.125000 ; 1 / 8  
 J[22][1] = 0.125000 ; 1 / 8  
 J[25][1] = 0.125000 ; 1 / 8  
 J[26][2] = 0.125000 ; 1 / 8  
 J[26][6] = 0.125000 ; 1 / 8  
 J[28][17] = 0.125000 ; 1 / 8  
 J[28][18] = 0.125000 ; 2 / 16  
 J[28][21] = 0.125000 ; 1 / 8  
 J[32][11] = 0.125000 ; 1 / 8  
 J[32][16] = 0.125000 ; 1 / 8  
 J[33][1] = 0.125000 ; 1 / 8  
 J[34][1] = 0.125000 ; 1 / 8  
 J[37][1] = 0.125000 ; 1 / 8  
 J[43][9] = 0.125000 ; 3 / 24  
 J[43][17] = 0.125000 ; 1 / 8  
 J[43][21] = 0.125000 ; 1 / 8  
 J[11][7] = 0.117647 ; 2 / 17  
 J[16][7] = 0.117647 ; 2 / 17  
 J[45][17] = 0.117647 ; 2 / 17  
 J[45][21] = 0.117647 ; 2 / 17  
 J[12][6] = 0.111111 ; 1 / 9  
 J[13][2] = 0.111111 ; 1 / 9  
 J[13][6] = 0.111111 ; 1 / 9  
 J[22][3] = 0.111111 ; 1 / 9  
 J[22][5] = 0.111111 ; 1 / 9  
 J[22][11] = 0.111111 ; 1 / 9  
 J[22][16] = 0.111111 ; 1 / 9

J[23][10] = 0.111111 ; 1 / 9  
 J[24][11] = 0.111111 ; 1 / 9  
 J[24][16] = 0.111111 ; 1 / 9  
 J[25][11] = 0.111111 ; 1 / 9  
 J[25][16] = 0.111111 ; 1 / 9  
 J[26][22] = 0.111111 ; 1 / 9  
 J[26][25] = 0.111111 ; 1 / 9  
 J[32][12] = 0.111111 ; 1 / 9  
 J[33][11] = 0.111111 ; 1 / 9  
 J[33][16] = 0.111111 ; 1 / 9  
 J[34][11] = 0.111111 ; 1 / 9  
 J[34][16] = 0.111111 ; 1 / 9  
 J[35][11] = 0.111111 ; 1 / 9  
 J[35][16] = 0.111111 ; 1 / 9  
 J[36][11] = 0.111111 ; 1 / 9  
 J[36][16] = 0.111111 ; 1 / 9  
 J[37][11] = 0.111111 ; 1 / 9  
 J[37][16] = 0.111111 ; 1 / 9  
 J[39][1] = 0.111111 ; 1 / 9  
 J[40][1] = 0.111111 ; 1 / 9  
 J[41][1] = 0.111111 ; 1 / 9  
 J[42][1] = 0.111111 ; 1 / 9  
 J[43][2] = 0.111111 ; 1 / 9  
 J[44][15] = 0.111111 ; 2 / 18  
 J[45][2] = 0.111111 ; 2 / 18  
 J[15][10] = 0.100000 ; 1 / 10  
 J[22][12] = 0.100000 ; 1 / 10  
 J[22][13] = 0.100000 ; 1 / 10  
 J[23][17] = 0.100000 ; 1 / 10  
 J[23][21] = 0.100000 ; 1 / 10  
 J[25][12] = 0.100000 ; 1 / 10  
 J[25][13] = 0.100000 ; 1 / 10  
 J[33][12] = 0.100000 ; 1 / 10  
 J[34][12] = 0.100000 ; 1 / 10  
 J[35][12] = 0.100000 ; 1 / 10  
 J[36][12] = 0.100000 ; 1 / 10  
 J[37][12] = 0.100000 ; 1 / 10  
 J[38][11] = 0.100000 ; 1 / 10  
 J[38][16] = 0.100000 ; 1 / 10  
 J[39][11] = 0.100000 ; 1 / 10  
 J[39][16] = 0.100000 ; 1 / 10  
 J[40][11] = 0.100000 ; 1 / 10  
 J[40][16] = 0.100000 ; 1 / 10  
 J[41][11] = 0.100000 ; 1 / 10  
 J[41][16] = 0.100000 ; 1 / 10  
 J[42][11] = 0.100000 ; 1 / 10  
 J[42][16] = 0.100000 ; 1 / 10  
 J[9][2] = 0.095238 ; 2 / 21  
 J[32][9] = 0.095238 ; 2 / 21  
 J[45][11] = 0.095238 ; 2 / 21  
 J[45][16] = 0.095238 ; 2 / 21

J[11][3] = 0.090909 ; 1 / 11  
 J[11][4] = 0.090909 ; 1 / 11  
 J[11][5] = 0.090909 ; 1 / 11  
 J[13][1] = 0.090909 ; 1 / 11  
 J[16][3] = 0.090909 ; 1 / 11  
 J[16][4] = 0.090909 ; 1 / 11  
 J[16][5] = 0.090909 ; 1 / 11  
 J[23][6] = 0.090909 ; 1 / 11  
 J[26][11] = 0.090909 ; 1 / 11  
 J[26][16] = 0.090909 ; 1 / 11  
 J[32][23] = 0.090909 ; 1 / 11  
 J[33][9] = 0.090909 ; 2 / 22  
 J[34][9] = 0.090909 ; 2 / 22  
 J[35][9] = 0.090909 ; 2 / 22  
 J[36][9] = 0.090909 ; 2 / 22  
 J[37][9] = 0.090909 ; 2 / 22  
 J[43][1] = 0.090909 ; 1 / 11  
 J[43][38] = 0.090909 ; 1 / 11  
 J[43][39] = 0.090909 ; 1 / 11  
 J[43][40] = 0.090909 ; 1 / 11  
 J[43][41] = 0.090909 ; 1 / 11  
 J[43][42] = 0.090909 ; 1 / 11  
 J[44][17] = 0.090909 ; 1 / 11  
 J[44][21] = 0.090909 ; 1 / 11  
 J[39][9] = 0.086957 ; 2 / 23  
 J[40][9] = 0.086957 ; 2 / 23  
 J[41][9] = 0.086957 ; 2 / 23  
 J[42][9] = 0.086957 ; 2 / 23  
 J[11][9] = 0.083333 ; 2 / 24  
 J[13][11] = 0.083333 ; 1 / 12  
 J[15][2] = 0.083333 ; 1 / 12  
 J[16][9] = 0.083333 ; 2 / 24  
 J[16][13] = 0.083333 ; 1 / 12  
 J[18][17] = 0.083333 ; 1 / 12  
 J[21][18] = 0.083333 ; 1 / 12  
 J[23][22] = 0.083333 ; 1 / 12  
 J[25][23] = 0.083333 ; 1 / 12  
 J[33][23] = 0.083333 ; 1 / 12  
 J[34][23] = 0.083333 ; 1 / 12  
 J[35][23] = 0.083333 ; 1 / 12  
 J[36][23] = 0.083333 ; 1 / 12  
 J[37][23] = 0.083333 ; 1 / 12  
 J[43][4] = 0.083333 ; 1 / 12  
 J[43][5] = 0.083333 ; 1 / 12  
 J[43][11] = 0.083333 ; 1 / 12  
 J[43][16] = 0.083333 ; 1 / 12  
 J[44][2] = 0.083333 ; 1 / 12  
 J[44][32] = 0.083333 ; 1 / 12  
 J[10][7] = 0.076923 ; 1 / 13  
 J[13][12] = 0.076923 ; 1 / 13  
 J[18][2] = 0.076923 ; 1 / 13

J[22][15] = 0.076923 ; 1 / 13  
 J[43][12] = 0.076923 ; 1 / 13  
 J[44][22] = 0.076923 ; 1 / 13  
 J[44][24] = 0.076923 ; 1 / 13  
 J[44][33] = 0.076923 ; 1 / 13  
 J[44][34] = 0.076923 ; 1 / 13  
 J[44][35] = 0.076923 ; 1 / 13  
 J[44][36] = 0.076923 ; 1 / 13  
 J[44][37] = 0.076923 ; 1 / 13  
 J[44][1] = 0.071429 ; 1 / 14  
 J[44][38] = 0.071429 ; 1 / 14  
 J[44][39] = 0.071429 ; 1 / 14  
 J[44][40] = 0.071429 ; 1 / 14  
 J[44][41] = 0.071429 ; 1 / 14  
 J[44][42] = 0.071429 ; 1 / 14  
 J[15][11] = 0.066667 ; 1 / 15  
 J[16][15] = 0.066667 ; 1 / 15  
 J[18][1] = 0.066667 ; 1 / 15  
 J[23][13] = 0.066667 ; 1 / 15  
 J[38][18] = 0.066667 ; 1 / 15  
 J[39][18] = 0.066667 ; 1 / 15  
 J[40][18] = 0.066667 ; 1 / 15  
 J[41][18] = 0.066667 ; 1 / 15  
 J[42][18] = 0.066667 ; 1 / 15  
 J[43][23] = 0.066667 ; 1 / 15  
 J[44][4] = 0.066667 ; 1 / 15  
 J[44][11] = 0.066667 ; 1 / 15  
 J[44][16] = 0.066667 ; 1 / 15  
 J[44][26] = 0.066667 ; 1 / 15  
 J[18][4] = 0.062500 ; 1 / 16  
 J[18][5] = 0.062500 ; 1 / 16  
 J[18][11] = 0.062500 ; 1 / 16  
 J[18][16] = 0.062500 ; 1 / 16  
 J[22][7] = 0.062500 ; 1 / 16  
 J[44][12] = 0.062500 ; 1 / 16  
 J[44][13] = 0.062500 ; 1 / 16  
 J[44][28] = 0.062500 ; 1 / 16  
 J[18][12] = 0.058824 ; 1 / 17  
 J[18][13] = 0.058824 ; 1 / 17  
 J[45][10] = 0.058824 ; 1 / 17  
 J[44][23] = 0.055556 ; 1 / 18  
 J[13][7] = 0.052632 ; 1 / 19  
 J[23][18] = 0.052632 ; 1 / 19  
 J[10][9] = 0.050000 ; 1 / 20  
 J[17][9] = 0.047619 ; 1 / 21  
 J[21][9] = 0.047619 ; 1 / 21  
 J[45][13] = 0.043478 ; 1 / 23  
 JLL OTHER EDGES = 0.000000

Node[1] = 16 , J[1] = 1.000000  
Node[2] = 11 , J[2] = 1.000000  
Node[3] = 21 , J[3] = 1.000000  
Node[4] = 17 , J[4] = 1.000000  
Node[5] = 44 , J[5] = 0.909091  
Node[6] = 18 , J[6] = 0.909091  
Node[7] = 38 , J[7] = 0.833333  
Node[8] = 4 , J[8] = 0.833333  
Node[9] = 7 , J[9] = 0.769231  
Node[10] = 45 , J[10] = 0.764706  
Node[11] = 9 , J[11] = 0.761905  
Node[12] = 25 , J[12] = 0.750000  
Node[13] = 6 , J[13] = 0.750000  
Node[14] = 32 , J[14] = 0.750000  
Node[15] = 24 , J[15] = 0.750000  
Node[16] = 33 , J[16] = 0.750000  
Node[17] = 34 , J[17] = 0.750000  
Node[18] = 35 , J[18] = 0.750000  
Node[19] = 36 , J[19] = 0.750000  
Node[20] = 37 , J[20] = 0.750000  
Node[21] = 26 , J[21] = 0.714286  
Node[22] = 5 , J[22] = 0.714286  
Node[23] = 3 , J[23] = 0.666667  
Node[24] = 39 , J[24] = 0.666667  
Node[25] = 40 , J[25] = 0.666667  
Node[26] = 41 , J[26] = 0.666667  
Node[27] = 42 , J[27] = 0.666667  
Node[28] = 2 , J[28] = 0.600000  
Node[29] = 1 , J[29] = 0.600000  
Node[30] = 15 , J[30] = 0.600000  
Node[31] = 28 , J[31] = 0.555556  
Node[32] = 12 , J[32] = 0.555556  
Node[33] = 10 , J[33] = 0.500000  
Node[34] = 43 , J[34] = 0.428571  
Node[35] = 22 , J[35] = 0.400000  
Node[36] = 23 , J[36] = 0.333333  
Node[37] = 13 , J[37] = 0.166667

**VERSION 4**  
**IDENTICAL**

I[14][13] = 0.700000 ; 7 / 10  
 I[15][1] = 0.500000 ; 6 / 12  
 I[26][24] = 0.400000 ; 2 / 5  
 I[9][5] = 0.250000 ; 1 / 4  
 I[15][8] = 0.214286 ; 3 / 14  
 I[8][3] = 0.200000 ; 3 / 15  
 I[7][5] = 0.166667 ; 1 / 6  
 I[9][7] = 0.166667 ; 1 / 6  
 I[12][5] = 0.166667 ; 1 / 6  
 I[12][9] = 0.166667 ; 1 / 6  
 I[24][18] = 0.166667 ; 1 / 6  
 I[12][4] = 0.142857 ; 2 / 14  
 I[24][19] = 0.142857 ; 1 / 7  
 I[8][5] = 0.125000 ; 1 / 8  
 I[9][8] = 0.125000 ; 1 / 8  
 I[10][5] = 0.125000 ; 1 / 8  
 I[10][7] = 0.125000 ; 1 / 8  
 I[10][3] = 0.125000 ; 1 / 8  
 I[12][7] = 0.125000 ; 1 / 8  
 I[16][4] = 0.125000 ; 2 / 16  
 I[19][18] = 0.125000 ; 1 / 8  
 I[26][18] = 0.125000 ; 1 / 8  
 I[3][2] = 0.117647 ; 2 / 17  
 I[8][7] = 0.111111 ; 1 / 9  
 I[26][19] = 0.111111 ; 1 / 9  
 I[15][3] = 0.105263 ; 2 / 19  
 I[12][8] = 0.100000 ; 1 / 10  
 I[12][10] = 0.100000 ; 1 / 10  
 I[10][8] = 0.090909 ; 1 / 11  
 I[18][16] = 0.090909 ; 1 / 11  
 I[24][21] = 0.090909 ; 1 / 11  
 I[21][18] = 0.083333 ; 1 / 12  
 I[24][16] = 0.083333 ; 1 / 12  
 I[5][3] = 0.076923 ; 1 / 13  
 I[9][3] = 0.076923 ; 1 / 13  
 I[12][3] = 0.076923 ; 1 / 13  
 I[16][12] = 0.076923 ; 1 / 13  
 I[18][2] = 0.076923 ; 1 / 13  
 I[19][2] = 0.076923 ; 1 / 13  
 I[21][19] = 0.076923 ; 1 / 13  
 I[24][2] = 0.076923 ; 1 / 13  
 I[26][21] = 0.076923 ; 1 / 13  
 I[5][4] = 0.071429 ; 1 / 14  
 I[7][3] = 0.071429 ; 1 / 14  
 I[7][4] = 0.071429 ; 1 / 14  
 I[9][4] = 0.071429 ; 1 / 14

I[19][16] = 0.071429 ; 1 / 14  
 I[26][16] = 0.071429 ; 1 / 14  
 I[10][3] = 0.066667 ; 1 / 15  
 I[26][2] = 0.066667 ; 1 / 15  
 I[10][4] = 0.062500 ; 1 / 16  
 I[21][16] = 0.062500 ; 1 / 16  
 I[8][4] = 0.058824 ; 1 / 17  
 I[16][2] = 0.058824 ; 1 / 17  
 I[21][2] = 0.058824 ; 1 / 17  
 I[4][3] = 0.052632 ; 1 / 19  
 I[11][2] = 0.047619 ; 1 / 21  
 I[11][3] = 0.047619 ; 1 / 21  
 JLL OTHER EDGES = 0.000000

Node[1] = 14 , I[1] = 0.700000  
 Node[2] = 13 , I[2] = 0.700000  
 Node[3] = 15 , I[3] = 0.500000  
 Node[4] = 1 , I[4] = 0.500000  
 Node[5] = 26 , I[5] = 0.400000  
 Node[6] = 24 , I[6] = 0.400000  
 Node[7] = 9 , I[7] = 0.250000  
 Node[8] = 5 , I[8] = 0.250000  
 Node[9] = 8 , I[9] = 0.214286  
 Node[10] = 3 , I[10] = 0.200000  
 Node[11] = 7 , I[11] = 0.166667  
 Node[12] = 12 , I[12] = 0.166667  
 Node[13] = 18 , I[13] = 0.166667  
 Node[14] = 4 , I[14] = 0.142857  
 Node[15] = 19 , I[15] = 0.142857  
 Node[16] = 10 , I[16] = 0.125000  
 Node[17] = 16 , I[17] = 0.125000  
 Node[18] = 2 , I[18] = 0.117647  
 Node[19] = 21 , I[19] = 0.090909  
 Node[20] = 11 , I[20] = 0.047619  
 Node[21] = 22 , I[21] = 0.000000

**VERSION 4  
COINCIDENTAL**

C[13][1] = 0.888889 ; 8 / 9  
 C[14][1] = 0.888889 ; 8 / 9  
 C[9][5] = 0.750000 ; 3 / 4  
 C[10][7] = 0.625000 ; 5 / 8  
 C[15][13] = 0.538462 ; 7 / 13  
 C[15][14] = 0.538462 ; 7 / 13  
 C[7][5] = 0.500000 ; 3 / 6  
 C[9][7] = 0.500000 ; 3 / 6  
 C[12][5] = 0.500000 ; 3 / 6  
 C[12][9] = 0.500000 ; 3 / 6  
 C[22][16] = 0.454545 ; 5 / 11  
 C[8][7] = 0.444444 ; 4 / 9  
 C[18][7] = 0.428571 ; 3 / 7  
 C[22][5] = 0.428571 ; 3 / 7  
 C[22][9] = 0.428571 ; 3 / 7  
 C[12][2] = 0.416667 ; 5 / 12  
 C[8][5] = 0.375000 ; 3 / 8  
 C[9][8] = 0.375000 ; 3 / 8  
 C[10][5] = 0.375000 ; 3 / 8  
 C[10][9] = 0.375000 ; 3 / 8  
 C[12][7] = 0.375000 ; 3 / 8  
 C[10][8] = 0.363636 ; 4 / 11  
 C[22][4] = 0.357143 ; 5 / 14  
 C[4][2] = 0.333333 ; 6 / 18  
 C[16][7] = 0.333333 ; 4 / 12  
 C[18][1] = 0.333333 ; 3 / 9  
 C[18][10] = 0.333333 ; 3 / 9  
 C[22][7] = 0.333333 ; 3 / 9  
 C[22][12] = 0.333333 ; 3 / 9  
 C[16][4] = 0.312500 ; 5 / 16  
 C[7][2] = 0.307692 ; 4 / 13  
 C[12][3] = 0.307692 ; 4 / 13  
 C[21][10] = 0.307692 ; 4 / 13  
 C[12][8] = 0.300000 ; 3 / 10  
 C[12][10] = 0.300000 ; 3 / 10  
 C[18][13] = 0.300000 ; 3 / 10  
 C[18][14] = 0.300000 ; 3 / 10  
 C[21][5] = 0.300000 ; 3 / 10  
 C[21][9] = 0.300000 ; 3 / 10  
 C[11][8] = 0.294118 ; 5 / 17  
 C[11][10] = 0.294118 ; 5 / 17  
 C[7][4] = 0.285714 ; 4 / 14  
 C[11][5] = 0.285714 ; 4 / 14  
 C[11][9] = 0.285714 ; 4 / 14  
 C[16][10] = 0.285714 ; 4 / 14  
 C[7][1] = 0.272727 ; 3 / 11

C[16][5] = 0.272727 ; 3 / 11  
 C[16][9] = 0.272727 ; 3 / 11  
 C[22][8] = 0.272727 ; 3 / 11  
 C[22][10] = 0.272727 ; 3 / 11  
 C[10][2] = 0.266667 ; 4 / 15  
 C[10][3] = 0.266667 ; 4 / 15  
 C[4][3] = 0.263158 ; 5 / 19  
 C[5][2] = 0.250000 ; 3 / 12  
 C[9][2] = 0.250000 ; 3 / 12  
 C[10][4] = 0.250000 ; 4 / 16  
 C[11][7] = 0.250000 ; 4 / 16  
 C[12][11] = 0.250000 ; 4 / 16  
 C[13][7] = 0.250000 ; 3 / 12  
 C[14][7] = 0.250000 ; 3 / 12  
 C[18][15] = 0.250000 ; 3 / 12  
 C[21][7] = 0.250000 ; 3 / 12  
 C[21][12] = 0.250000 ; 3 / 12  
 C[22][18] = 0.250000 ; 2 / 8  
 C[22][21] = 0.250000 ; 3 / 12  
 C[24][3] = 0.250000 ; 3 / 12  
 C[3][2] = 0.235294 ; 4 / 17  
 C[10][1] = 0.230769 ; 3 / 13  
 C[16][3] = 0.222222 ; 4 / 18  
 C[24][10] = 0.222222 ; 2 / 9  
 C[7][3] = 0.214286 ; 3 / 14  
 C[12][4] = 0.214286 ; 3 / 14  
 C[13][10] = 0.214286 ; 3 / 14  
 C[14][10] = 0.214286 ; 3 / 14  
 C[15][7] = 0.214286 ; 3 / 14  
 C[18][4] = 0.214286 ; 3 / 14  
 C[21][8] = 0.214286 ; 3 / 14  
 C[22][2] = 0.214286 ; 3 / 14  
 C[26][3] = 0.214286 ; 3 / 14  
 C[21][11] = 0.210526 ; 4 / 19  
 C[16][8] = 0.200000 ; 3 / 15  
 C[18][8] = 0.200000 ; 2 / 10  
 C[22][3] = 0.200000 ; 3 / 15  
 C[26][24] = 0.200000 ; 1 / 5  
 C[11][3] = 0.190476 ; 4 / 21  
 C[8][2] = 0.187500 ; 3 / 16  
 C[15][10] = 0.187500 ; 3 / 16  
 C[19][11] = 0.187500 ; 3 / 16  
 C[18][16] = 0.181818 ; 2 / 11  
 C[26][10] = 0.181818 ; 2 / 11  
 C[8][4] = 0.176471 ; 3 / 17  
 C[16][2] = 0.176471 ; 3 / 17  
 C[22][11] = 0.176471 ; 3 / 17  
 C[4][1] = 0.166667 ; 3 / 18  
 C[21][3] = 0.166667 ; 3 / 18  
 C[24][5] = 0.166667 ; 1 / 6  
 C[24][9] = 0.166667 ; 1 / 6

C[13][4] = 0.157895 ; 3 / 19  
 C[14][4] = 0.157895 ; 3 / 19  
 C[21][4] = 0.157895 ; 3 / 19  
 C[5][3] = 0.153846 ; 2 / 13  
 C[9][3] = 0.153846 ; 2 / 13  
 C[16][12] = 0.153846 ; 2 / 13  
 C[19][2] = 0.153846 ; 2 / 13  
 C[5][4] = 0.142857 ; 2 / 14  
 C[8][1] = 0.142857 ; 2 / 14  
 C[9][4] = 0.142857 ; 2 / 14  
 C[11][2] = 0.142857 ; 3 / 21  
 C[15][4] = 0.142857 ; 3 / 21  
 C[15][8] = 0.142857 ; 2 / 14  
 C[16][11] = 0.142857 ; 3 / 21  
 C[18][3] = 0.142857 ; 2 / 14  
 C[18][5] = 0.142857 ; 1 / 7  
 C[18][9] = 0.142857 ; 1 / 7  
 C[8][3] = 0.133333 ; 2 / 15  
 C[13][8] = 0.133333 ; 2 / 15  
 C[14][8] = 0.133333 ; 2 / 15  
 C[24][11] = 0.133333 ; 2 / 15  
 C[11][4] = 0.125000 ; 3 / 24  
 C[16][1] = 0.125000 ; 2 / 16  
 C[19][5] = 0.125000 ; 1 / 8  
 C[19][9] = 0.125000 ; 1 / 8  
 C[21][16] = 0.125000 ; 2 / 16  
 C[24][7] = 0.125000 ; 1 / 8  
 C[24][12] = 0.125000 ; 1 / 8  
 C[24][22] = 0.125000 ; 1 / 8  
 C[26][5] = 0.125000 ; 1 / 8  
 C[26][9] = 0.125000 ; 1 / 8  
 C[2][1] = 0.117647 ; 2 / 17  
 C[16][13] = 0.117647 ; 2 / 17  
 C[16][14] = 0.117647 ; 2 / 17  
 C[21][2] = 0.117647 ; 2 / 17  
 C[26][11] = 0.117647 ; 2 / 17  
 C[3][1] = 0.111111 ; 2 / 18  
 C[13][2] = 0.111111 ; 2 / 18  
 C[14][2] = 0.111111 ; 2 / 18  
 C[18][12] = 0.111111 ; 1 / 9  
 C[13][3] = 0.105263 ; 2 / 19  
 C[14][3] = 0.105263 ; 2 / 19  
 C[15][3] = 0.105263 ; 2 / 19  
 C[16][15] = 0.105263 ; 2 / 19  
 C[14][13] = 0.100000 ; 1 / 10  
 C[15][2] = 0.100000 ; 2 / 20  
 C[19][7] = 0.100000 ; 1 / 10  
 C[19][12] = 0.100000 ; 1 / 10  
 C[22][19] = 0.100000 ; 1 / 10  
 C[24][1] = 0.100000 ; 1 / 10  
 C[24][8] = 0.100000 ; 1 / 10

C[26][7] = 0.100000 ; 1 / 10  
 C[26][12] = 0.100000 ; 1 / 10  
 C[26][22] = 0.100000 ; 1 / 10  
 C[5][1] = 0.090909 ; 1 / 11  
 C[9][1] = 0.090909 ; 1 / 11  
 C[24][13] = 0.090909 ; 1 / 11  
 C[24][14] = 0.090909 ; 1 / 11  
 C[15][11] = 0.086957 ; 2 / 23  
 C[13][5] = 0.083333 ; 1 / 12  
 C[13][9] = 0.083333 ; 1 / 12  
 C[14][5] = 0.083333 ; 1 / 12  
 C[14][9] = 0.083333 ; 1 / 12  
 C[15][1] = 0.083333 ; 1 / 12  
 C[19][1] = 0.083333 ; 1 / 12  
 C[19][8] = 0.083333 ; 1 / 12  
 C[19][10] = 0.083333 ; 1 / 12  
 C[26][1] = 0.083333 ; 1 / 12  
 C[26][8] = 0.083333 ; 1 / 12  
 C[12][1] = 0.076923 ; 1 / 13  
 C[18][2] = 0.076923 ; 1 / 13  
 C[19][13] = 0.076923 ; 1 / 13  
 C[19][14] = 0.076923 ; 1 / 13  
 C[22][1] = 0.076923 ; 1 / 13  
 C[24][15] = 0.076923 ; 1 / 13  
 C[26][13] = 0.076923 ; 1 / 13  
 C[26][14] = 0.076923 ; 1 / 13  
 C[13][12] = 0.071429 ; 1 / 14  
 C[14][12] = 0.071429 ; 1 / 14  
 C[15][5] = 0.071429 ; 1 / 14  
 C[15][9] = 0.071429 ; 1 / 14  
 C[22][13] = 0.071429 ; 1 / 14  
 C[22][14] = 0.071429 ; 1 / 14  
 C[19][15] = 0.066667 ; 1 / 15  
 C[24][4] = 0.066667 ; 1 / 15  
 C[26][15] = 0.066667 ; 1 / 15  
 C[15][12] = 0.062500 ; 1 / 16  
 C[19][3] = 0.062500 ; 1 / 16  
 C[21][1] = 0.062500 ; 1 / 16  
 C[22][15] = 0.062500 ; 1 / 16  
 C[18][11] = 0.058824 ; 1 / 17  
 C[19][4] = 0.058824 ; 1 / 17  
 C[21][13] = 0.058824 ; 1 / 17  
 C[21][14] = 0.058824 ; 1 / 17  
 C[26][4] = 0.058824 ; 1 / 17  
 C[21][15] = 0.052632 ; 1 / 19  
 C[11][1] = 0.047619 ; 1 / 21  
 C[13][11] = 0.045455 ; 1 / 22  
 C[14][11] = 0.045455 ; 1 / 22  
 JLL OTHER EDGES = 0.000000



Node[1] = 13 , C[1] = 0.888889  
Node[2] = 1 , C[2] = 0.888889  
Node[3] = 14 , C[3] = 0.888889  
Node[4] = 9 , C[4] = 0.750000  
Node[5] = 5 , C[5] = 0.750000  
Node[6] = 10 , C[6] = 0.625000  
Node[7] = 7 , C[7] = 0.625000  
Node[8] = 15 , C[8] = 0.538462  
Node[9] = 12 , C[9] = 0.500000  
Node[10] = 22 , C[10] = 0.454545  
Node[11] = 16 , C[11] = 0.454545  
Node[12] = 8 , C[12] = 0.444444  
Node[13] = 18 , C[13] = 0.428571  
Node[14] = 2 , C[14] = 0.416667  
Node[15] = 4 , C[15] = 0.357143  
Node[16] = 3 , C[16] = 0.307692  
Node[17] = 21 , C[17] = 0.307692  
Node[18] = 11 , C[18] = 0.294118  
Node[19] = 24 , C[19] = 0.250000  
Node[20] = 26 , C[20] = 0.214286  
Node[21] = 19 , C[21] = 0.187500

**VERSION 4  
COMPOSITE**

J[9][5] = 1.000000 ; 4 / 4  
J[13][1] = 0.888889 ; 8 / 9  
J[14][1] = 0.888889 ; 8 / 9  
J[14][13] = 0.800000 ; 8 / 10  
J[10][7] = 0.750000 ; 6 / 8  
J[7][5] = 0.666667 ; 4 / 6  
J[9][7] = 0.666667 ; 4 / 6  
J[12][5] = 0.666667 ; 4 / 6  
J[12][9] = 0.666667 ; 4 / 6  
J[26][24] = 0.600000 ; 3 / 5  
J[15][1] = 0.583333 ; 7 / 12  
J[8][7] = 0.555556 ; 5 / 9  
J[15][13] = 0.538462 ; 7 / 13  
J[15][14] = 0.538462 ; 7 / 13  
J[8][5] = 0.500000 ; 4 / 8  
J[9][8] = 0.500000 ; 4 / 8  
J[10][5] = 0.500000 ; 4 / 8  
J[10][9] = 0.500000 ; 4 / 8  
J[12][7] = 0.500000 ; 4 / 8  
J[10][8] = 0.454545 ; 5 / 11  
J[22][16] = 0.454545 ; 5 / 11  
J[16][4] = 0.437500 ; 7 / 16  
J[18][7] = 0.428571 ; 3 / 7  
J[22][5] = 0.428571 ; 3 / 7  
J[22][9] = 0.428571 ; 3 / 7  
J[12][2] = 0.416667 ; 5 / 12  
J[12][8] = 0.400000 ; 4 / 10  
J[12][10] = 0.400000 ; 4 / 10  
J[12][3] = 0.384615 ; 5 / 13  
J[7][4] = 0.357143 ; 5 / 14  
J[12][4] = 0.357143 ; 5 / 14  
J[15][8] = 0.357143 ; 5 / 14  
J[22][4] = 0.357143 ; 5 / 14  
J[3][2] = 0.352941 ; 6 / 17  
J[4][2] = 0.333333 ; 6 / 18  
J[8][3] = 0.333333 ; 5 / 15  
J[10][3] = 0.333333 ; 5 / 15  
J[16][7] = 0.333333 ; 4 / 12  
J[18][1] = 0.333333 ; 3 / 9  
J[18][10] = 0.333333 ; 3 / 9  
J[22][7] = 0.333333 ; 3 / 9  
J[22][12] = 0.333333 ; 3 / 9  
J[4][3] = 0.315789 ; 6 / 19  
J[10][4] = 0.312500 ; 5 / 16  
J[7][2] = 0.307692 ; 4 / 13

J[21][10] = 0.307692 ; 4 / 13  
J[18][13] = 0.300000 ; 3 / 10  
J[18][14] = 0.300000 ; 3 / 10  
J[21][5] = 0.300000 ; 3 / 10  
J[21][9] = 0.300000 ; 3 / 10  
J[11][8] = 0.294118 ; 5 / 17  
J[11][10] = 0.294118 ; 5 / 17  
J[7][3] = 0.285714 ; 4 / 14  
J[11][5] = 0.285714 ; 4 / 14  
J[11][9] = 0.285714 ; 4 / 14  
J[16][10] = 0.285714 ; 4 / 14  
J[7][1] = 0.272727 ; 3 / 11  
J[16][5] = 0.272727 ; 3 / 11  
J[16][9] = 0.272727 ; 3 / 11  
J[18][16] = 0.272727 ; 3 / 11  
J[22][8] = 0.272727 ; 3 / 11  
J[22][10] = 0.272727 ; 3 / 11  
J[10][2] = 0.266667 ; 4 / 15  
J[5][2] = 0.250000 ; 3 / 12  
J[9][2] = 0.250000 ; 3 / 12  
J[11][7] = 0.250000 ; 4 / 16  
J[12][11] = 0.250000 ; 4 / 16  
J[13][7] = 0.250000 ; 3 / 12  
J[14][7] = 0.250000 ; 3 / 12  
J[18][15] = 0.250000 ; 3 / 12  
J[21][7] = 0.250000 ; 3 / 12  
J[21][12] = 0.250000 ; 3 / 12  
J[22][18] = 0.250000 ; 2 / 8  
J[22][21] = 0.250000 ; 3 / 12  
J[24][3] = 0.250000 ; 3 / 12  
J[11][3] = 0.238095 ; 5 / 21  
J[8][4] = 0.235294 ; 4 / 17  
J[16][2] = 0.235294 ; 4 / 17  
J[5][3] = 0.230769 ; 3 / 13  
J[9][3] = 0.230769 ; 3 / 13  
J[10][1] = 0.230769 ; 3 / 13  
J[16][12] = 0.230769 ; 3 / 13  
J[19][2] = 0.230769 ; 3 / 13  
J[16][3] = 0.222222 ; 4 / 18  
J[24][10] = 0.222222 ; 2 / 9  
J[5][4] = 0.214286 ; 3 / 14  
J[9][4] = 0.214286 ; 3 / 14  
J[13][10] = 0.214286 ; 3 / 14  
J[14][10] = 0.214286 ; 3 / 14  
J[15][7] = 0.214286 ; 3 / 14  
J[18][4] = 0.214286 ; 3 / 14  
J[21][8] = 0.214286 ; 3 / 14  
J[22][2] = 0.214286 ; 3 / 14  
J[26][3] = 0.214286 ; 3 / 14  
J[15][3] = 0.210526 ; 4 / 19  
J[21][11] = 0.210526 ; 4 / 19

J[16][8] = 0.200000 ; 3 / 15  
 J[18][8] = 0.200000 ; 2 / 10  
 J[22][3] = 0.200000 ; 3 / 15  
 J[11][2] = 0.190476 ; 4 / 21  
 J[8][2] = 0.187500 ; 3 / 16  
 J[15][10] = 0.187500 ; 3 / 16  
 J[19][11] = 0.187500 ; 3 / 16  
 J[21][16] = 0.187500 ; 3 / 16  
 J[26][10] = 0.181818 ; 2 / 11  
 J[21][2] = 0.176471 ; 3 / 17  
 J[22][11] = 0.176471 ; 3 / 17  
 J[4][1] = 0.166667 ; 3 / 18  
 J[21][3] = 0.166667 ; 3 / 18  
 J[24][5] = 0.166667 ; 1 / 6  
 J[24][9] = 0.166667 ; 1 / 6  
 J[24][18] = 0.166667 ; 1 / 6  
 J[13][4] = 0.157895 ; 3 / 19  
 J[14][4] = 0.157895 ; 3 / 19  
 J[21][4] = 0.157895 ; 3 / 19  
 J[18][2] = 0.153846 ; 2 / 13  
 J[8][1] = 0.142857 ; 2 / 14  
 J[15][4] = 0.142857 ; 3 / 21  
 J[16][11] = 0.142857 ; 3 / 21  
 J[18][3] = 0.142857 ; 2 / 14  
 J[18][5] = 0.142857 ; 1 / 7  
 J[18][9] = 0.142857 ; 1 / 7  
 J[24][19] = 0.142857 ; 1 / 7  
 J[13][8] = 0.133333 ; 2 / 15  
 J[14][8] = 0.133333 ; 2 / 15  
 J[24][11] = 0.133333 ; 2 / 15  
 J[11][4] = 0.125000 ; 3 / 24  
 J[16][1] = 0.125000 ; 2 / 16  
 J[19][5] = 0.125000 ; 1 / 8  
 J[19][9] = 0.125000 ; 1 / 8  
 J[19][18] = 0.125000 ; 1 / 8  
 J[24][7] = 0.125000 ; 1 / 8  
 J[24][12] = 0.125000 ; 1 / 8  
 J[24][22] = 0.125000 ; 1 / 8  
 J[26][5] = 0.125000 ; 1 / 8  
 J[26][9] = 0.125000 ; 1 / 8  
 J[26][18] = 0.125000 ; 1 / 8  
 J[2][1] = 0.117647 ; 2 / 17  
 J[16][13] = 0.117647 ; 2 / 17  
 J[16][14] = 0.117647 ; 2 / 17  
 J[26][11] = 0.117647 ; 2 / 17  
 J[3][1] = 0.111111 ; 2 / 18  
 J[13][2] = 0.111111 ; 2 / 18  
 J[14][2] = 0.111111 ; 2 / 18  
 J[18][12] = 0.111111 ; 1 / 9  
 J[26][19] = 0.111111 ; 1 / 9  
 J[13][3] = 0.105263 ; 2 / 19

J[14][3] = 0.105263 ; 2 / 19  
 J[16][15] = 0.105263 ; 2 / 19  
 J[15][2] = 0.100000 ; 2 / 20  
 J[19][7] = 0.100000 ; 1 / 10  
 J[19][12] = 0.100000 ; 1 / 10  
 J[22][19] = 0.100000 ; 1 / 10  
 J[24][1] = 0.100000 ; 1 / 10  
 J[24][8] = 0.100000 ; 1 / 10  
 J[26][7] = 0.100000 ; 1 / 10  
 J[26][12] = 0.100000 ; 1 / 10  
 J[26][22] = 0.100000 ; 1 / 10  
 J[5][1] = 0.090909 ; 1 / 11  
 J[9][1] = 0.090909 ; 1 / 11  
 J[24][13] = 0.090909 ; 1 / 11  
 J[24][14] = 0.090909 ; 1 / 11  
 J[24][21] = 0.090909 ; 1 / 11  
 J[15][11] = 0.086957 ; 2 / 23  
 J[13][5] = 0.083333 ; 1 / 12  
 J[13][9] = 0.083333 ; 1 / 12  
 J[14][5] = 0.083333 ; 1 / 12  
 J[14][9] = 0.083333 ; 1 / 12  
 J[19][1] = 0.083333 ; 1 / 12  
 J[19][8] = 0.083333 ; 1 / 12  
 J[19][10] = 0.083333 ; 1 / 12  
 J[21][18] = 0.083333 ; 1 / 12  
 J[24][16] = 0.083333 ; 1 / 12  
 J[26][1] = 0.083333 ; 1 / 12  
 J[26][8] = 0.083333 ; 1 / 12  
 J[12][1] = 0.076923 ; 1 / 13  
 J[19][13] = 0.076923 ; 1 / 13  
 J[19][14] = 0.076923 ; 1 / 13  
 J[21][19] = 0.076923 ; 1 / 13  
 J[22][1] = 0.076923 ; 1 / 13  
 J[24][2] = 0.076923 ; 1 / 13  
 J[24][15] = 0.076923 ; 1 / 13  
 J[26][13] = 0.076923 ; 1 / 13  
 J[26][14] = 0.076923 ; 1 / 13  
 J[26][21] = 0.076923 ; 1 / 13  
 J[13][12] = 0.071429 ; 1 / 14  
 J[14][12] = 0.071429 ; 1 / 14  
 J[15][5] = 0.071429 ; 1 / 14  
 J[15][9] = 0.071429 ; 1 / 14  
 J[19][16] = 0.071429 ; 1 / 14  
 J[22][13] = 0.071429 ; 1 / 14  
 J[22][14] = 0.071429 ; 1 / 14  
 J[26][16] = 0.071429 ; 1 / 14  
 J[19][15] = 0.066667 ; 1 / 15  
 J[24][4] = 0.066667 ; 1 / 15  
 J[26][2] = 0.066667 ; 1 / 15  
 J[26][15] = 0.066667 ; 1 / 15  
 J[15][12] = 0.062500 ; 1 / 16

```

J[19][3] = 0.062500 ; 1 / 16
J[21][1] = 0.062500 ; 1 / 16
J[22][15] = 0.062500 ; 1 / 16
J[18][11] = 0.058824 ; 1 / 17
J[19][4] = 0.058824 ; 1 / 17
J[21][13] = 0.058824 ; 1 / 17
J[21][14] = 0.058824 ; 1 / 17
J[26][4] = 0.058824 ; 1 / 17
J[21][15] = 0.052632 ; 1 / 19
J[11][1] = 0.047619 ; 1 / 21
J[13][11] = 0.045455 ; 1 / 22
J[14][11] = 0.045455 ; 1 / 22
JLL OTHER EDGES = 0.000000

```

```

Node[1] = 9 , J[1] = 1.000000
Node[2] = 5 , J[2] = 1.000000
Node[3] = 13 , J[3] = 0.888889
Node[4] = 1 , J[4] = 0.888889
Node[5] = 14 , J[5] = 0.888889
Node[6] = 10 , J[6] = 0.750000
Node[7] = 7 , J[7] = 0.750000
Node[8] = 12 , J[8] = 0.666667
Node[9] = 26 , J[9] = 0.600000
Node[10] = 24 , J[10] = 0.600000
Node[11] = 15 , J[11] = 0.583333
Node[12] = 8 , J[12] = 0.555556
Node[13] = 22 , J[13] = 0.454545
Node[14] = 16 , J[14] = 0.454545
Node[15] = 4 , J[15] = 0.437500
Node[16] = 18 , J[16] = 0.428571
Node[17] = 2 , J[17] = 0.416667
Node[18] = 3 , J[18] = 0.384615
Node[19] = 21 , J[19] = 0.307692
Node[20] = 11 , J[20] = 0.294118
Node[21] = 19 , J[21] = 0.230769

```

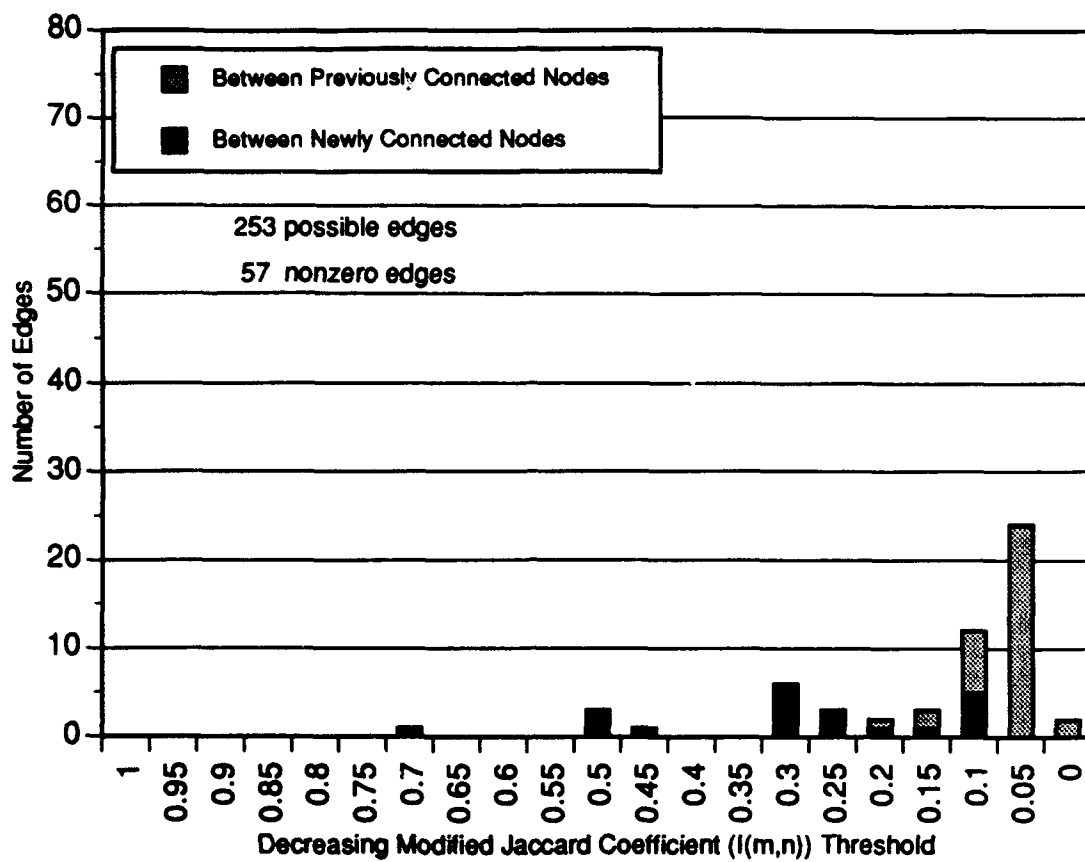
## **APPENIX F**

### **HISTOGRAMS**

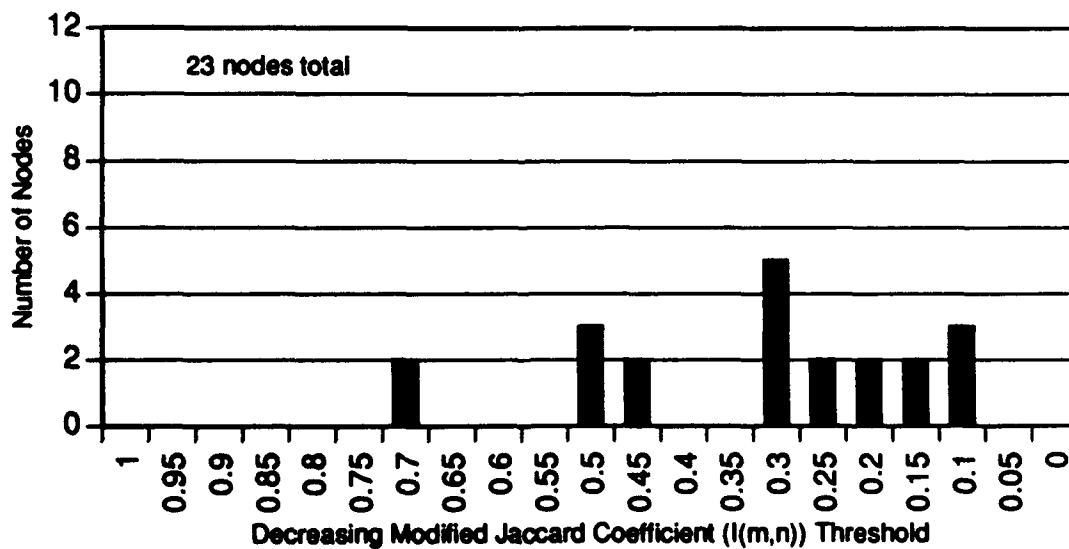
There is one figure in this appendix for each dimension of each experimental and random version. Each figure contains two histograms. The first histogram shows how many edges are added to the graph in each interval of the Jaccard coefficient. In the second histogram, the column in each Jaccard coefficient interval shows how many nodes have their largest incident edge in that interval.

In the first histogram, the total column height in each interval shows the number of edges that have weights in that interval. The column is divided into two parts. The black part, labeled "Between Newly Connected Nodes," shows the numbers of edges that are incident on nodes that had no incident edge in a higher threshold interval. The gray part, labeled "Between Previously Connected Nodes," shows the numbers of edges that are incident on nodes that did have an incident edge in a higher threshold interval.

The abscissae of the histograms are labeled with the Jaccard coefficient decreasing from left to right. The histograms are divided into intervals of 0.05. In general, the data included in an interval are strictly less than the upper limit of the interval and greater than or equal to the lower limit. There are two exceptions: data in the uppermost interval are less than or equal to unity; data in the lowermost interval are strictly greater than zero.

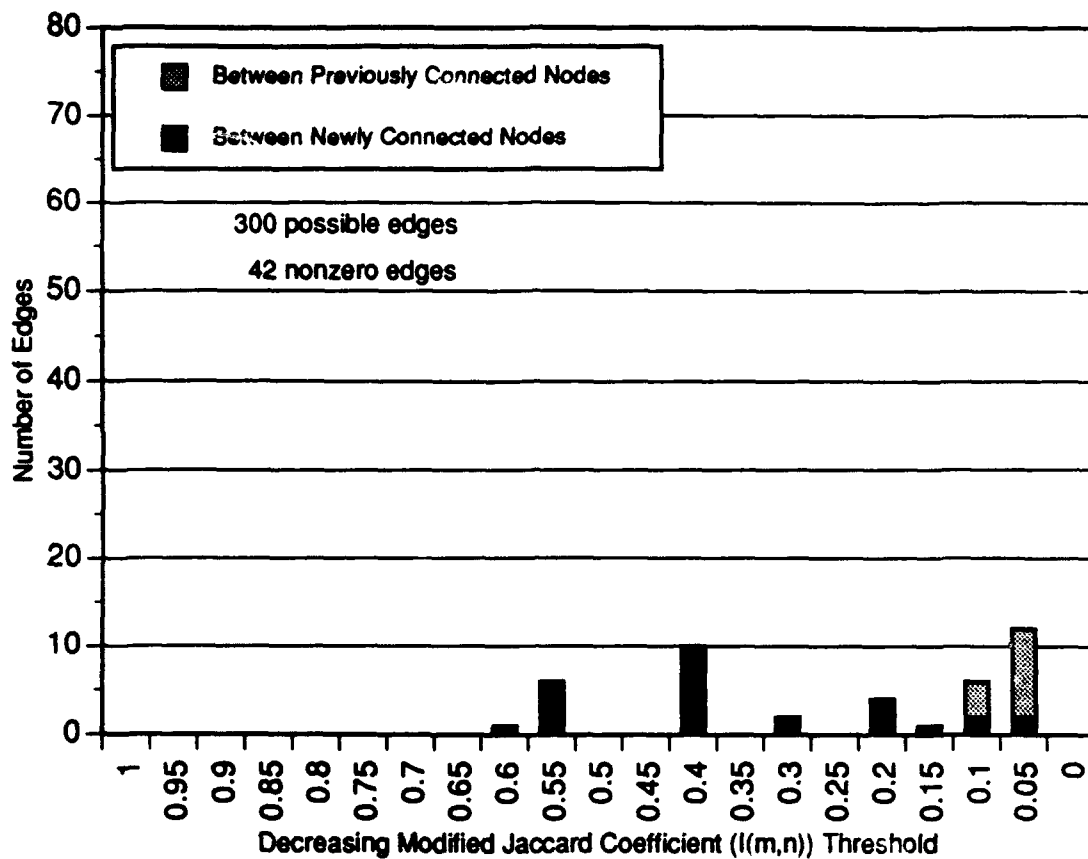


(a) Edges Added at Each Threshold

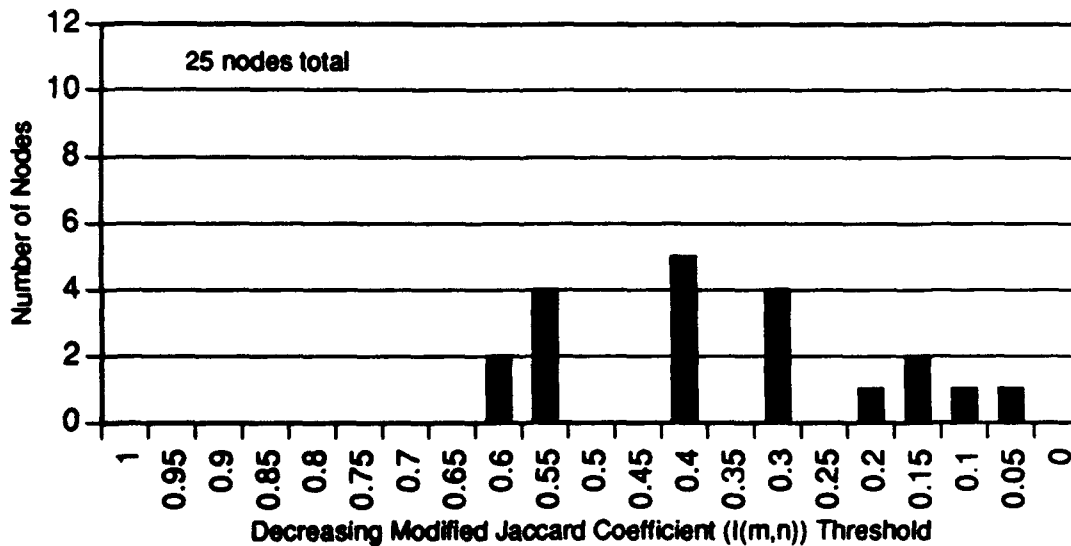


(b) Newly Connected Nodes at Each Threshold

Figure F.1 - Version 1, Identical Bounds

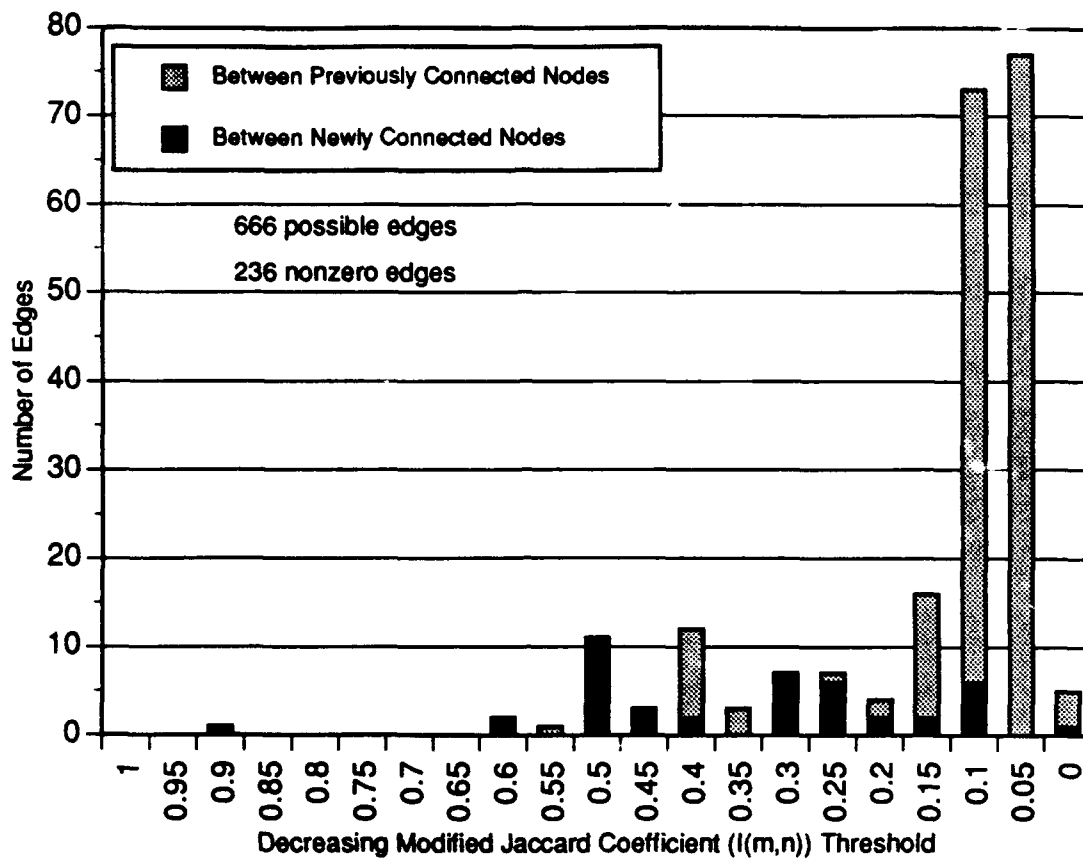


(a) Edges Added at Each Threshold

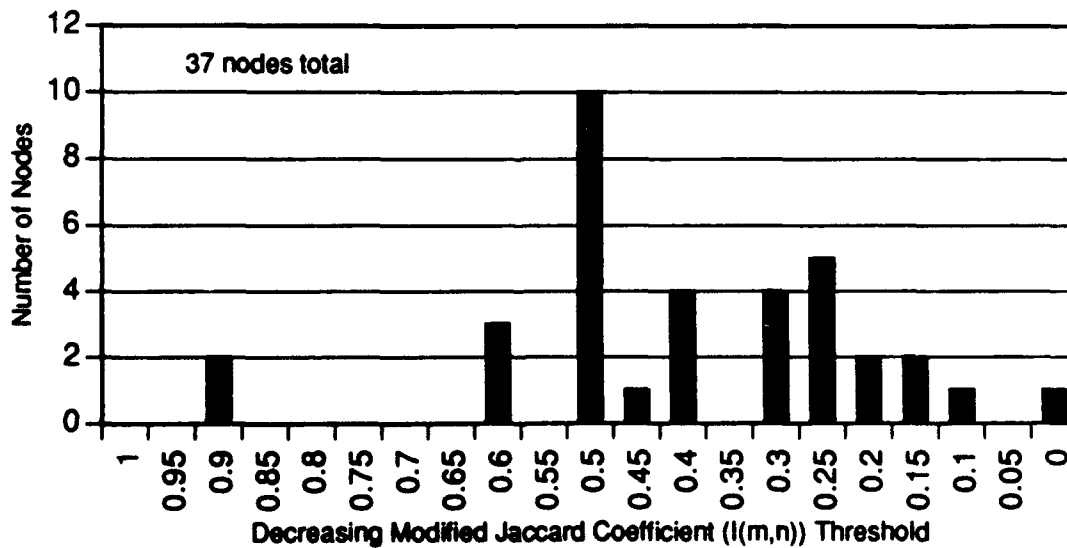


(b) Newly Connected Nodes at Each Threshold

Figure F.2 - Version 2, Identical Bounds



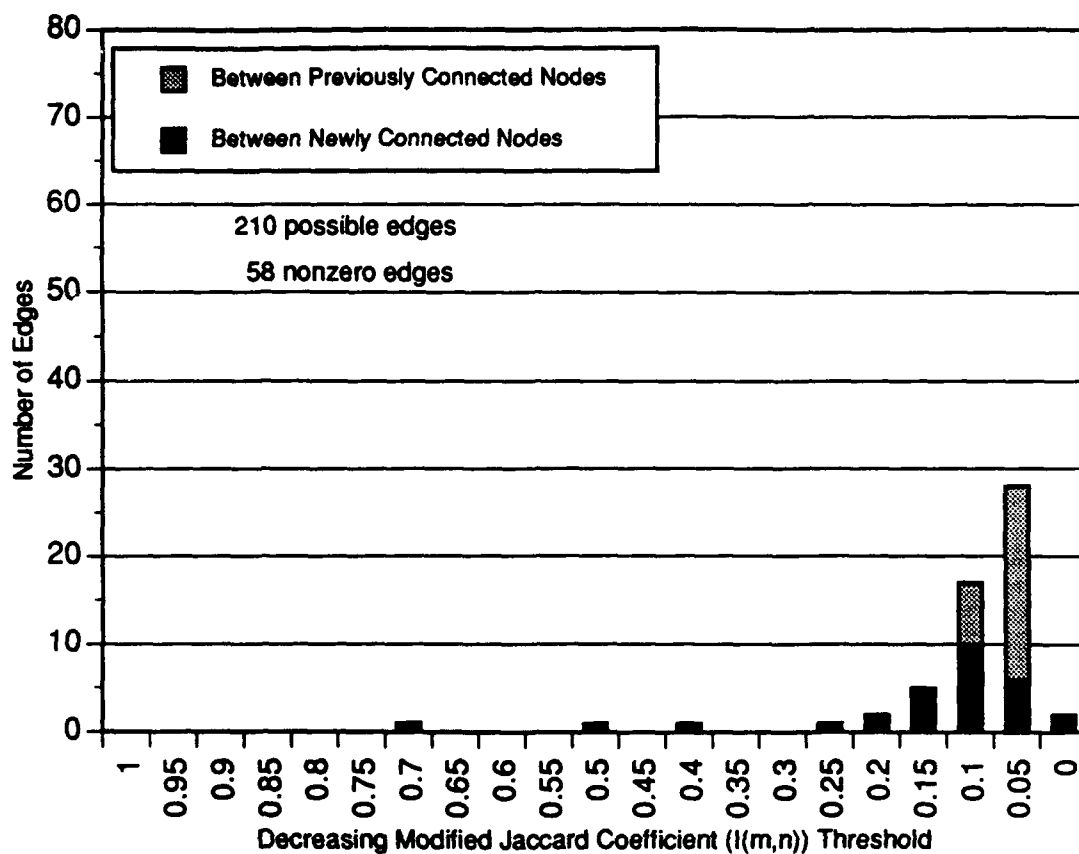
(a) Edges Added at Each Threshold



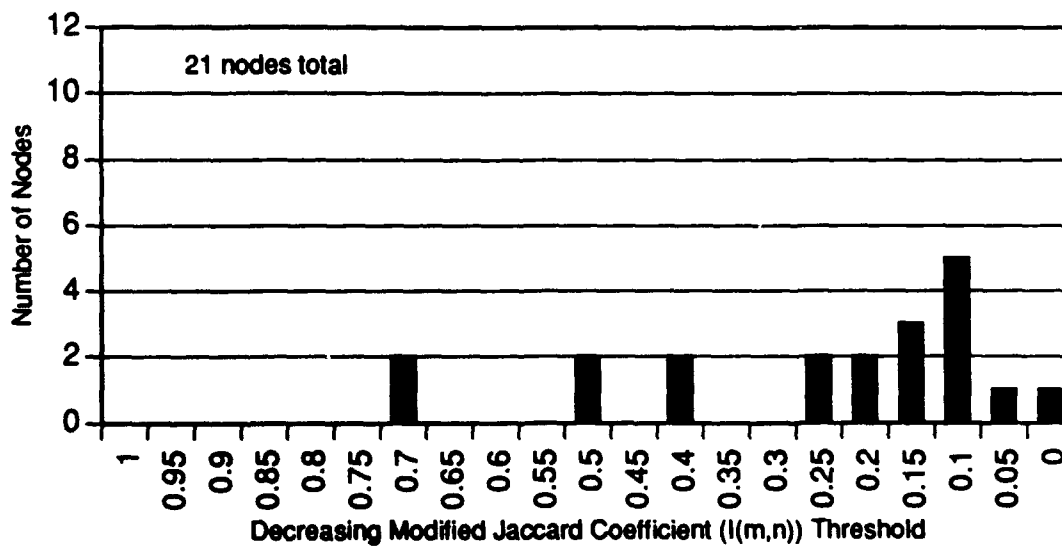
(b) Newly Connected Nodes at Each Threshold

Figure F.3 - Version 3, Identical Bounds



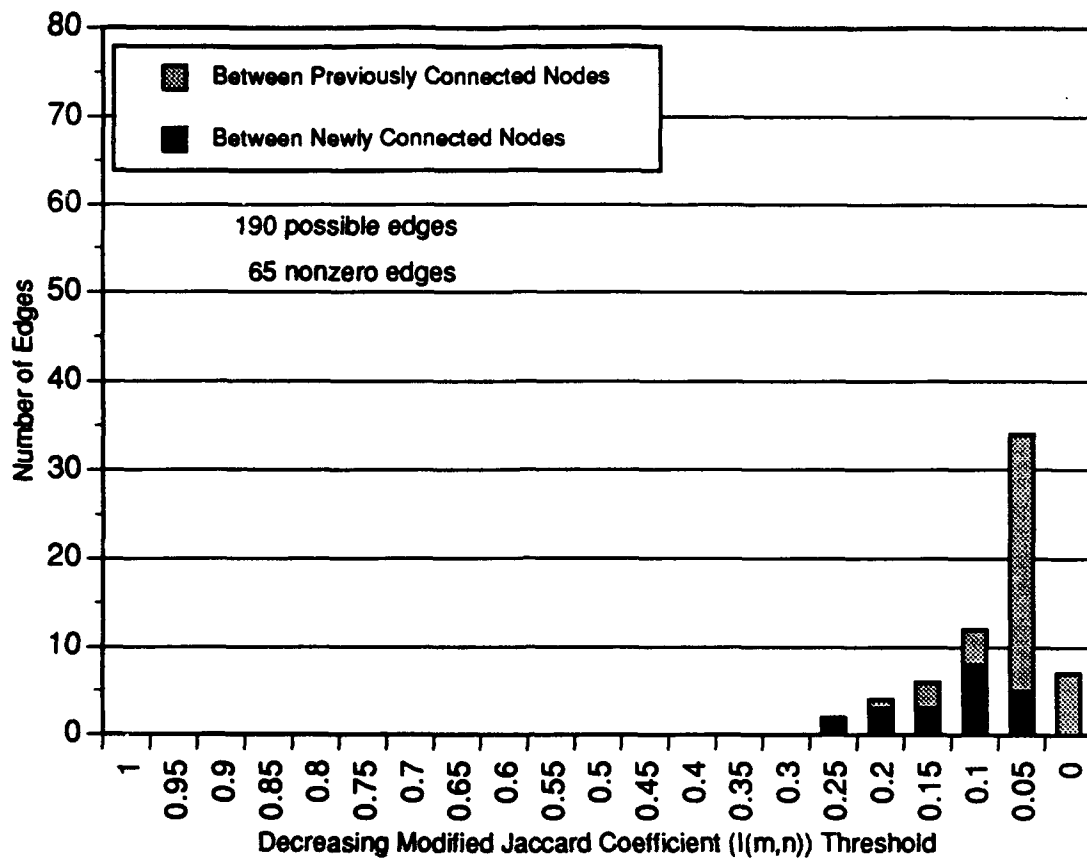


(a) Edges Added at Each Threshold

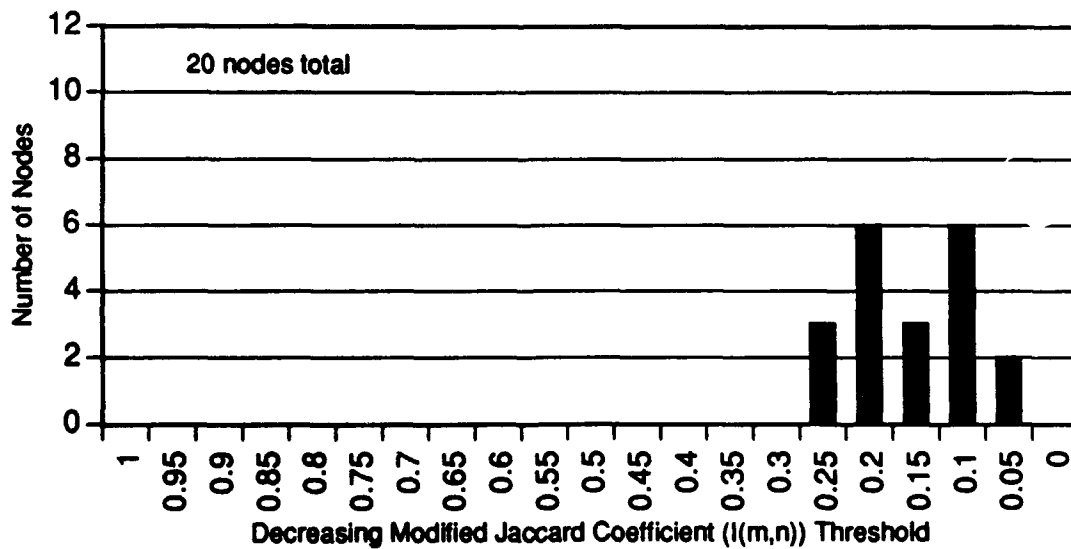


(b) Newly Connected Nodes at Each Threshold

Figure F.4 - Version 4, Identical Bounds

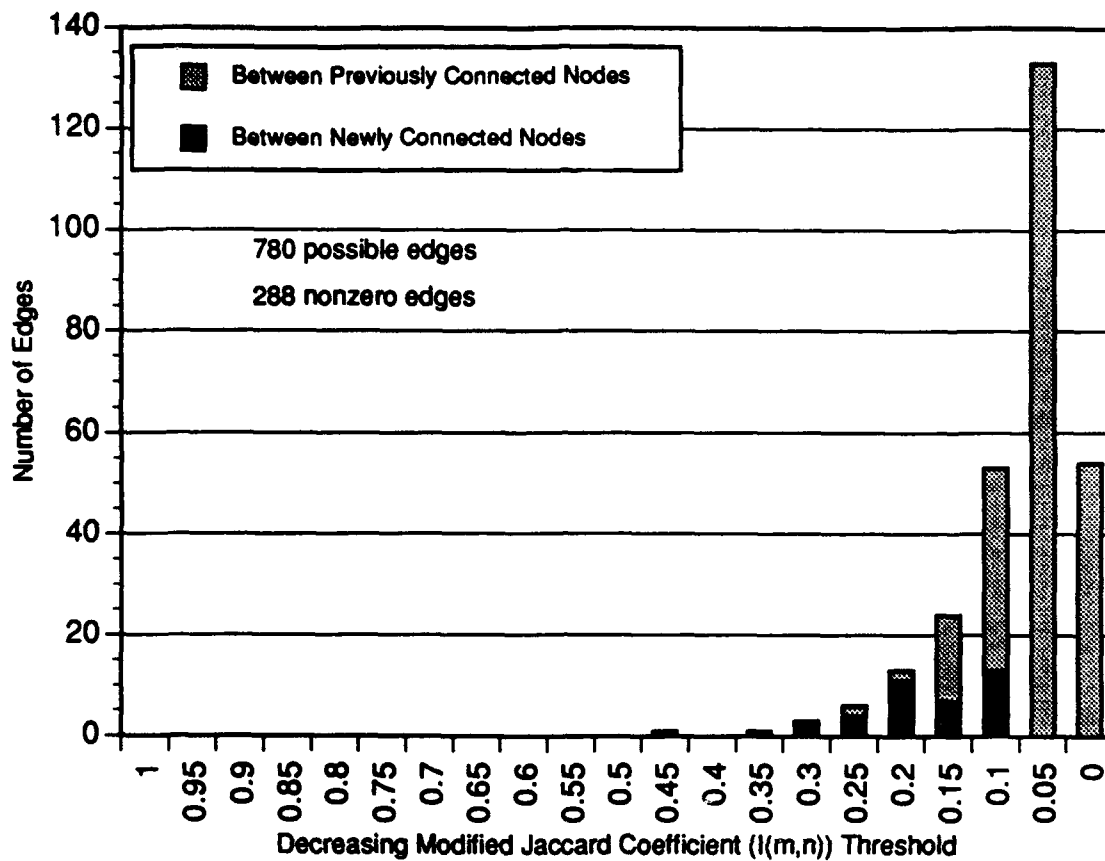


(a) Edges Added at Each Threshold

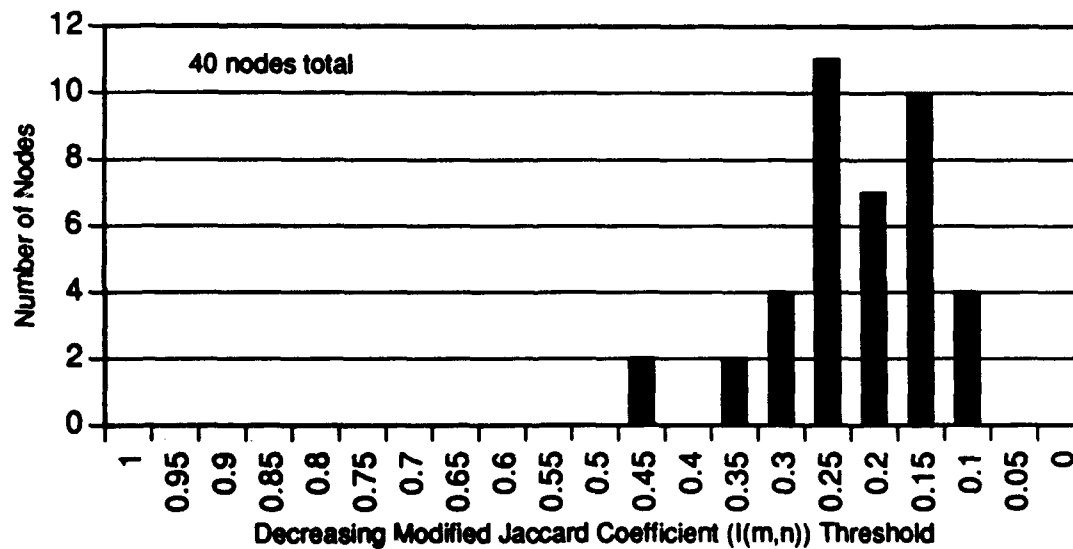


(b) Newly Connected Nodes at Each Threshold

Figure F.5 - Version R20, Identical Bounds

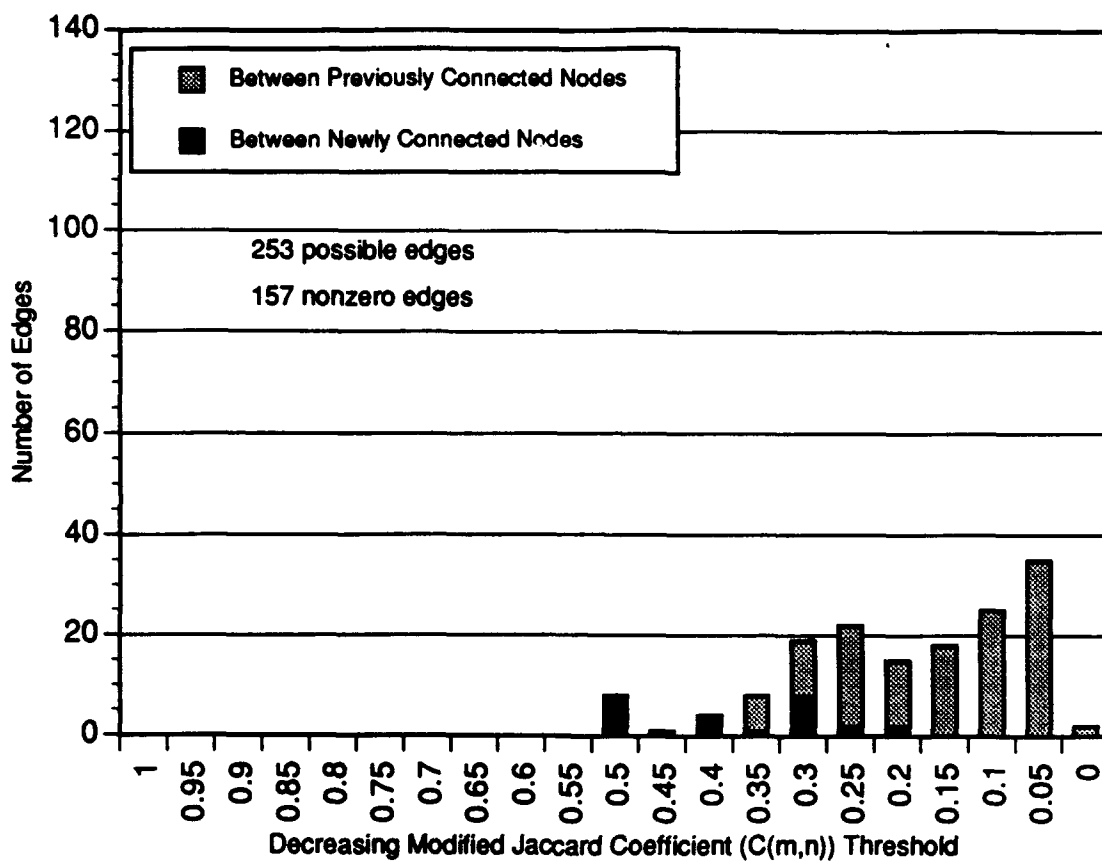


(a) Edges Added at Each Threshold

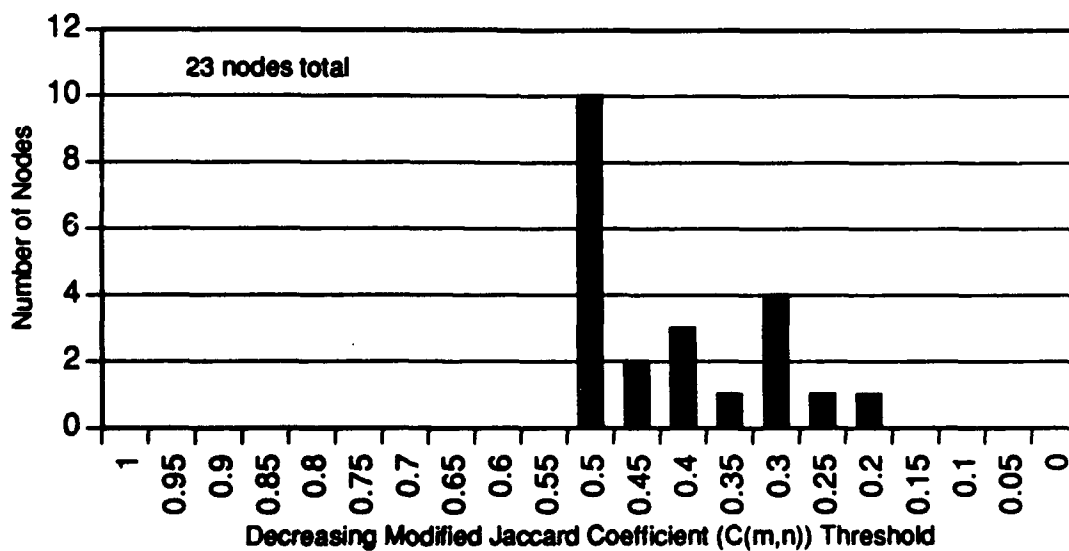


(b) Newly Connected Nodes at Each Threshold

Figure F.6 - Version R40, Identical Bounds

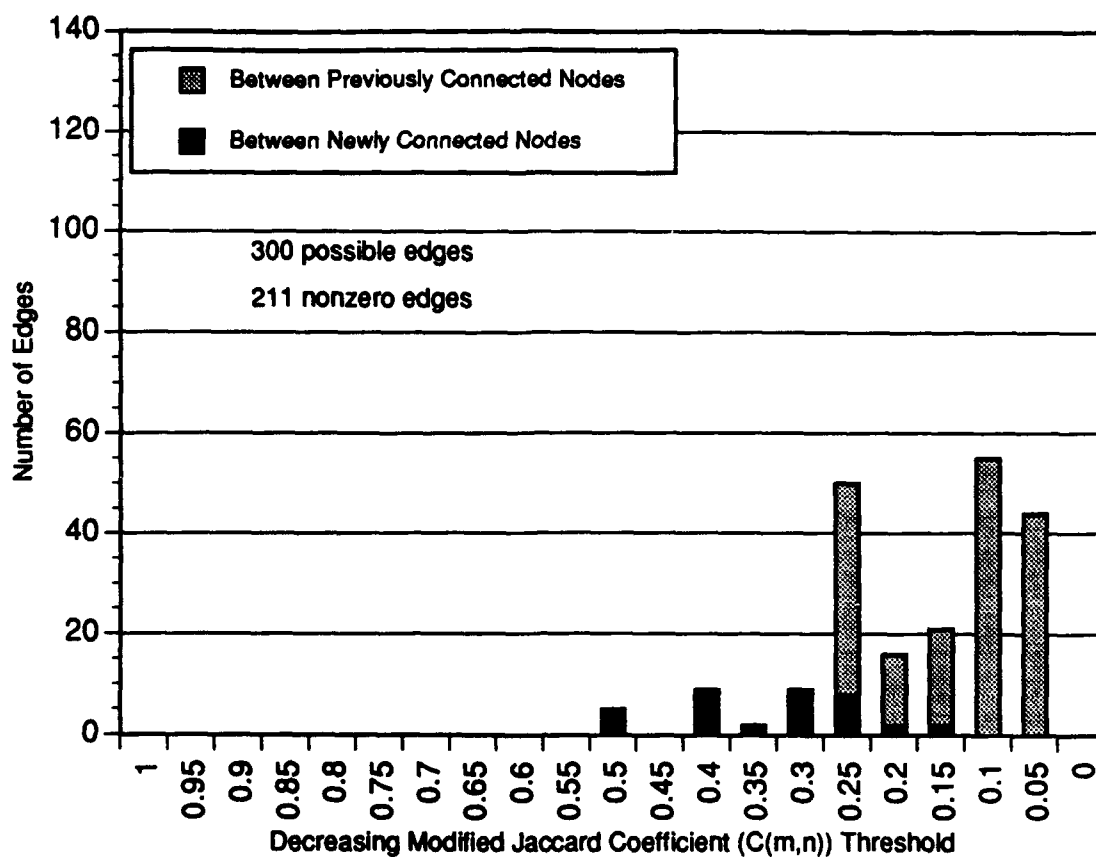


(a) Edges Added at Each Threshold

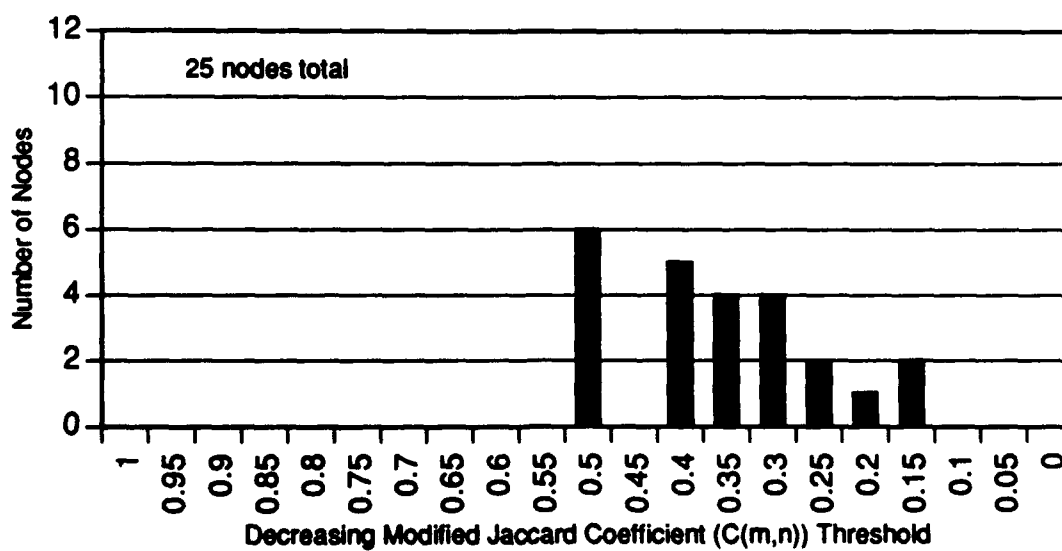


(b) Newly Connected Nodes at Each Threshold

**Figure F.7 - Version 1, Coincidental Bounds**

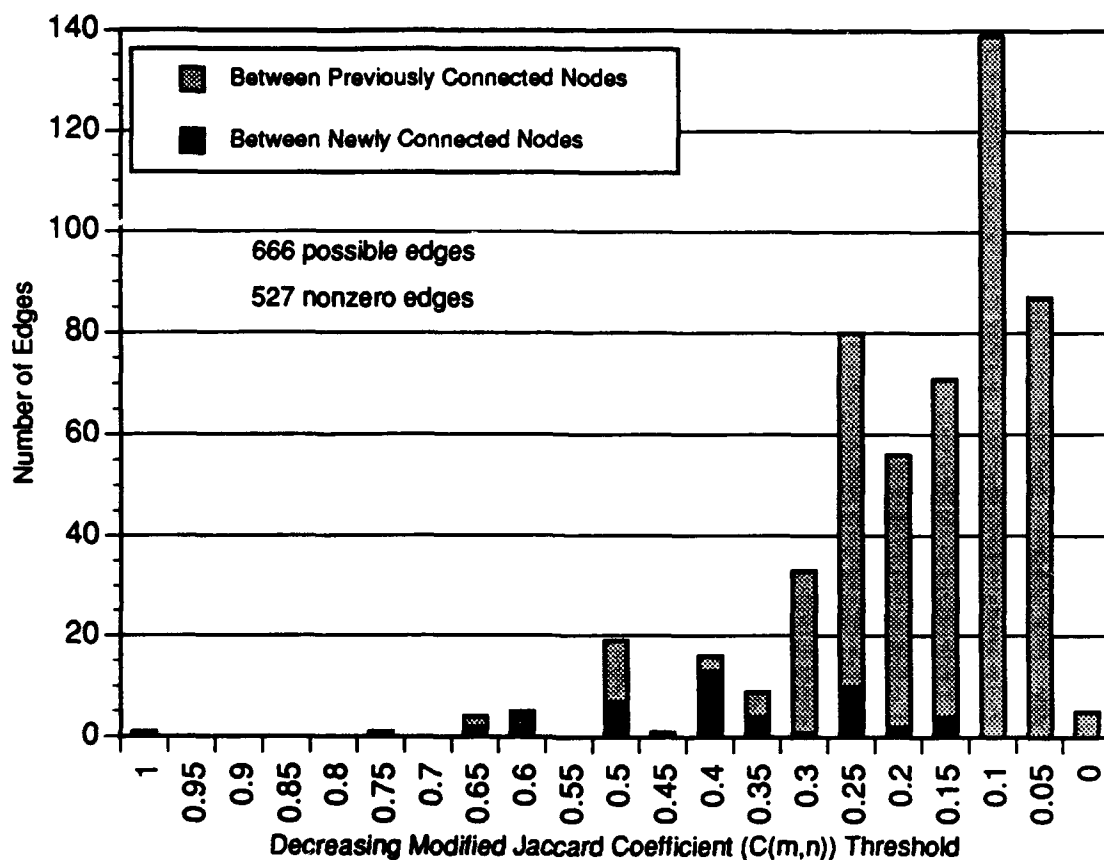


(a) Edges Added at Each Threshold

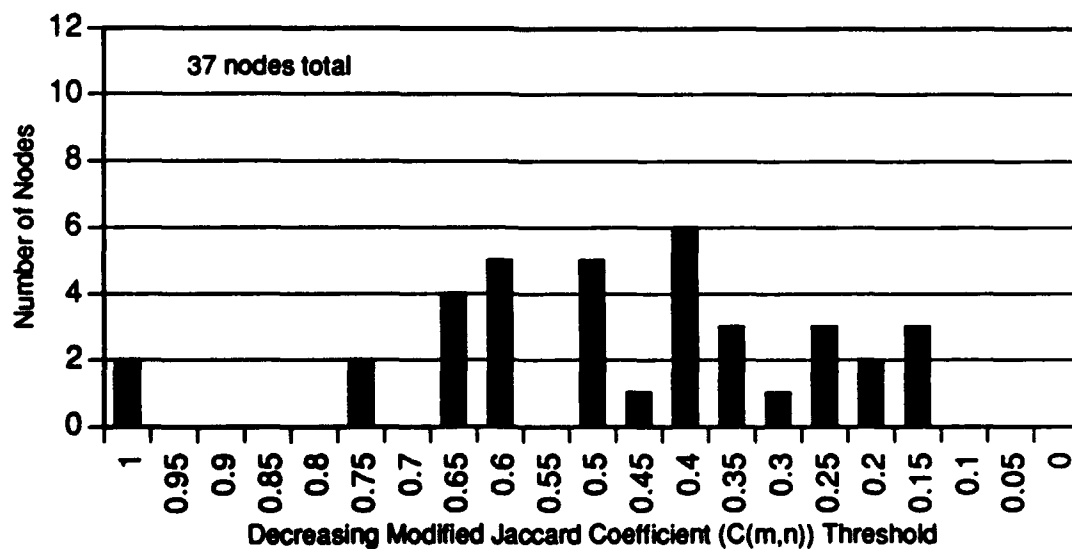


(b) Newly Connected Nodes at Each Threshold

Figure F.8 - Version 2, Coincidental Bounds

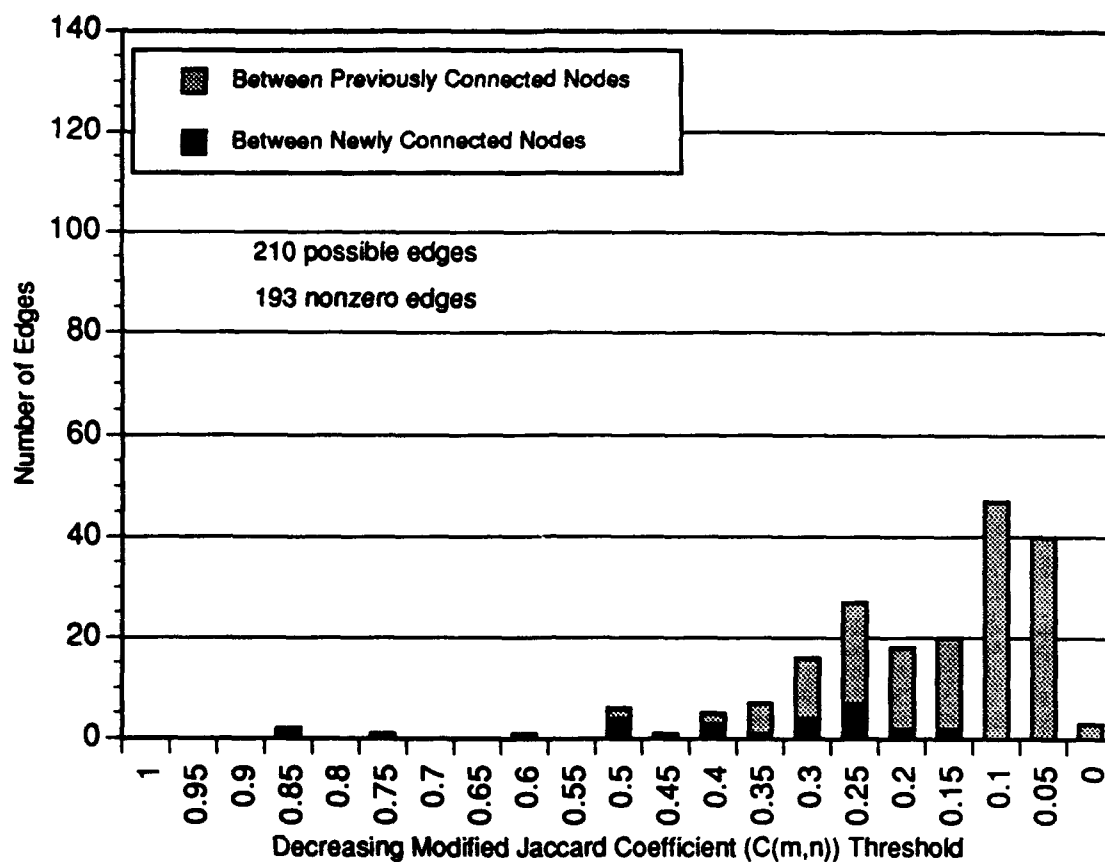


(a) Edges Added at Each Threshold

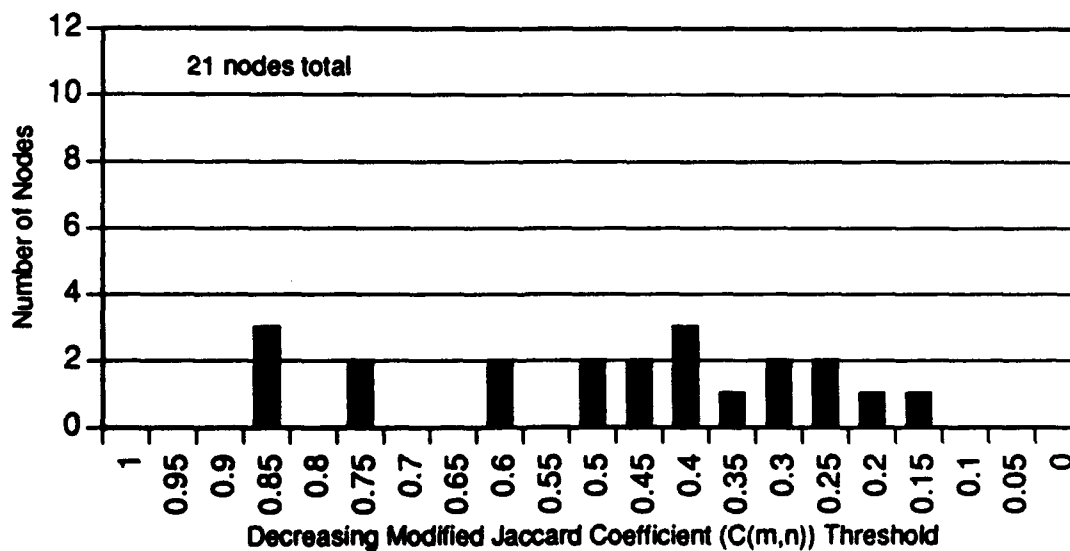


(b) Newly Connected Nodes at Each Threshold

**Figure F.9 - Version 3, Coincidental Bounds**

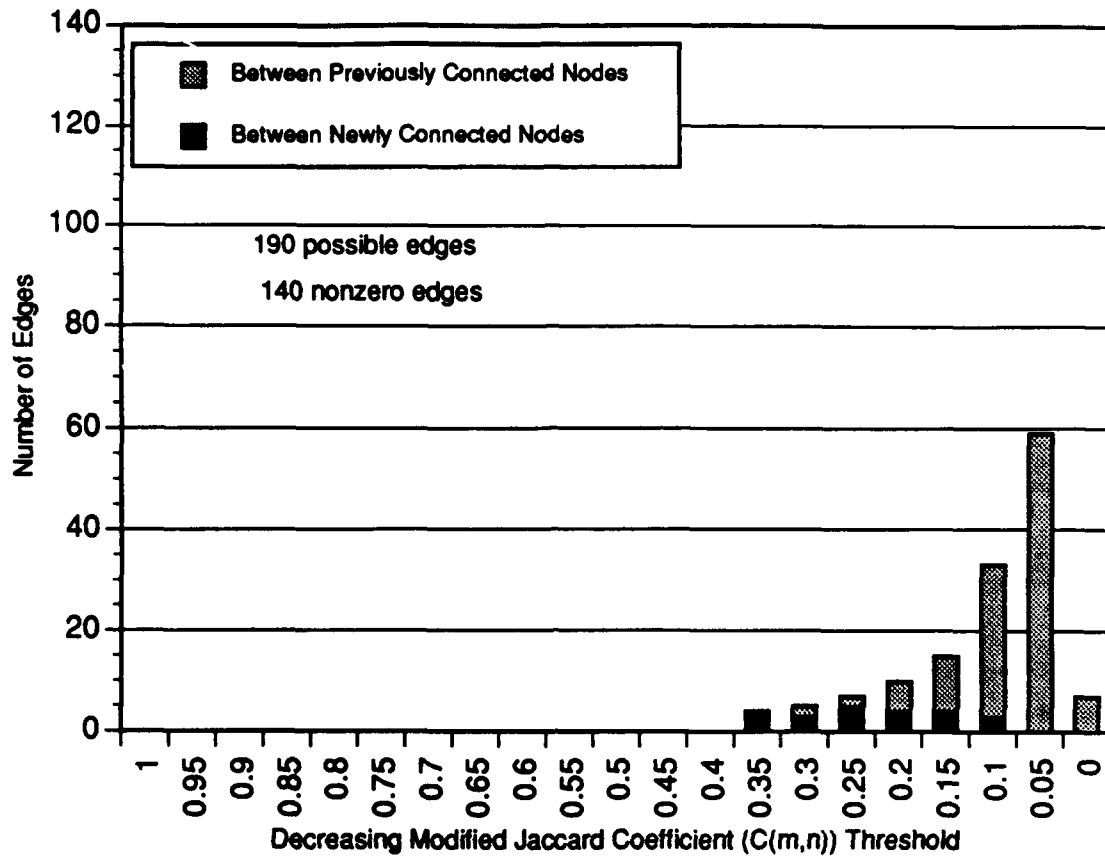


(a) Edges Added at Each Threshold

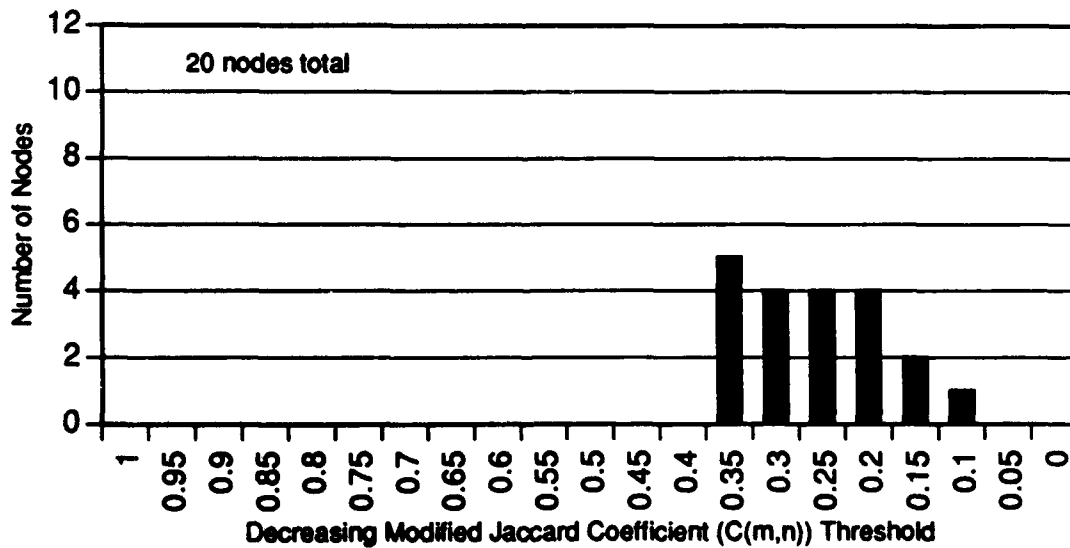


(b) Newly Connected Nodes at Each Threshold

Figure F.10 - Version 4, Coincidental Bounds



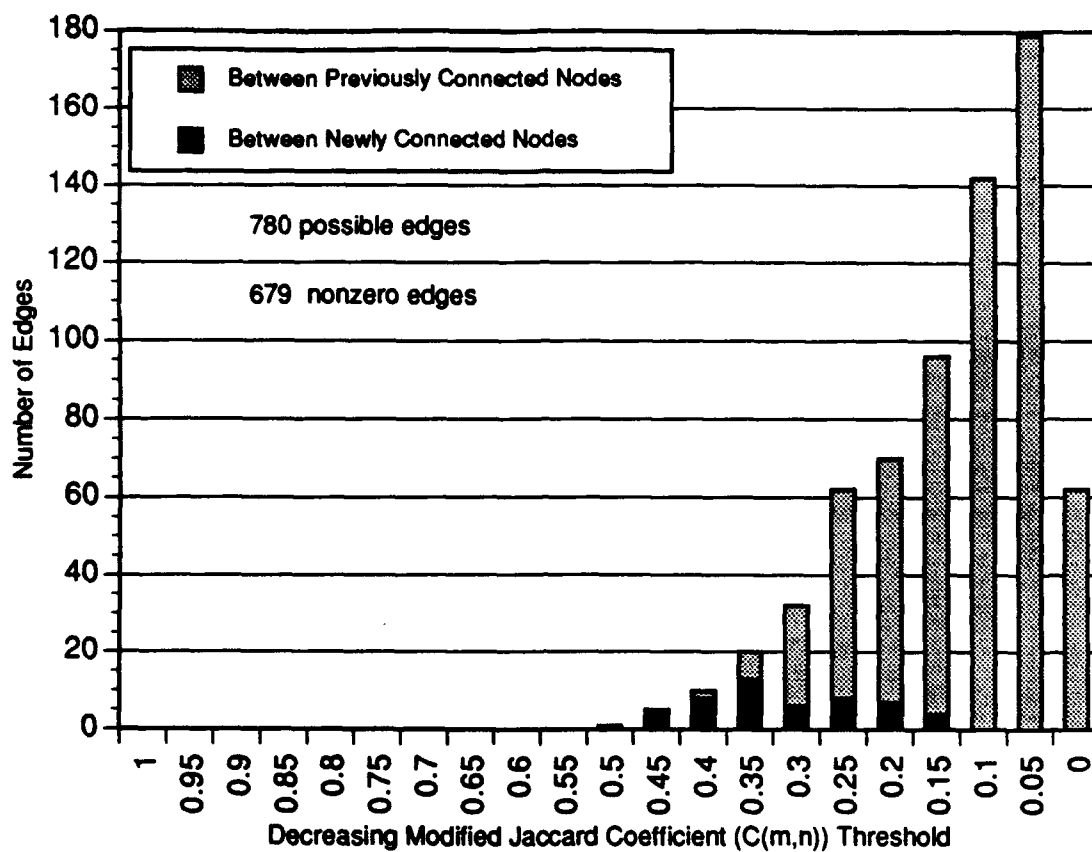
(a) Edges Added at Each Threshold



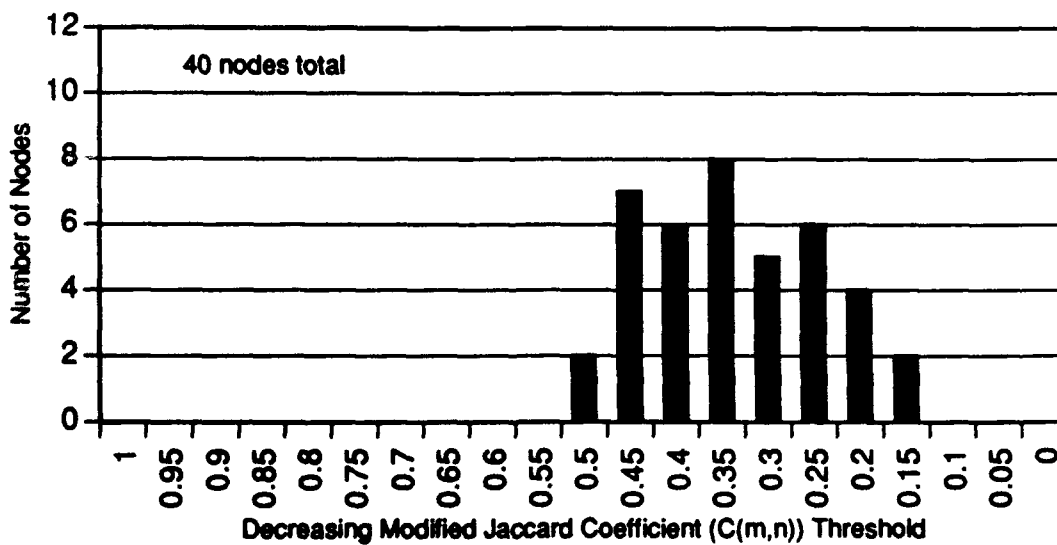
(b) Newly Connected Nodes at Each Threshold

Figure F.11 - Version R20, Coincidental Bounds



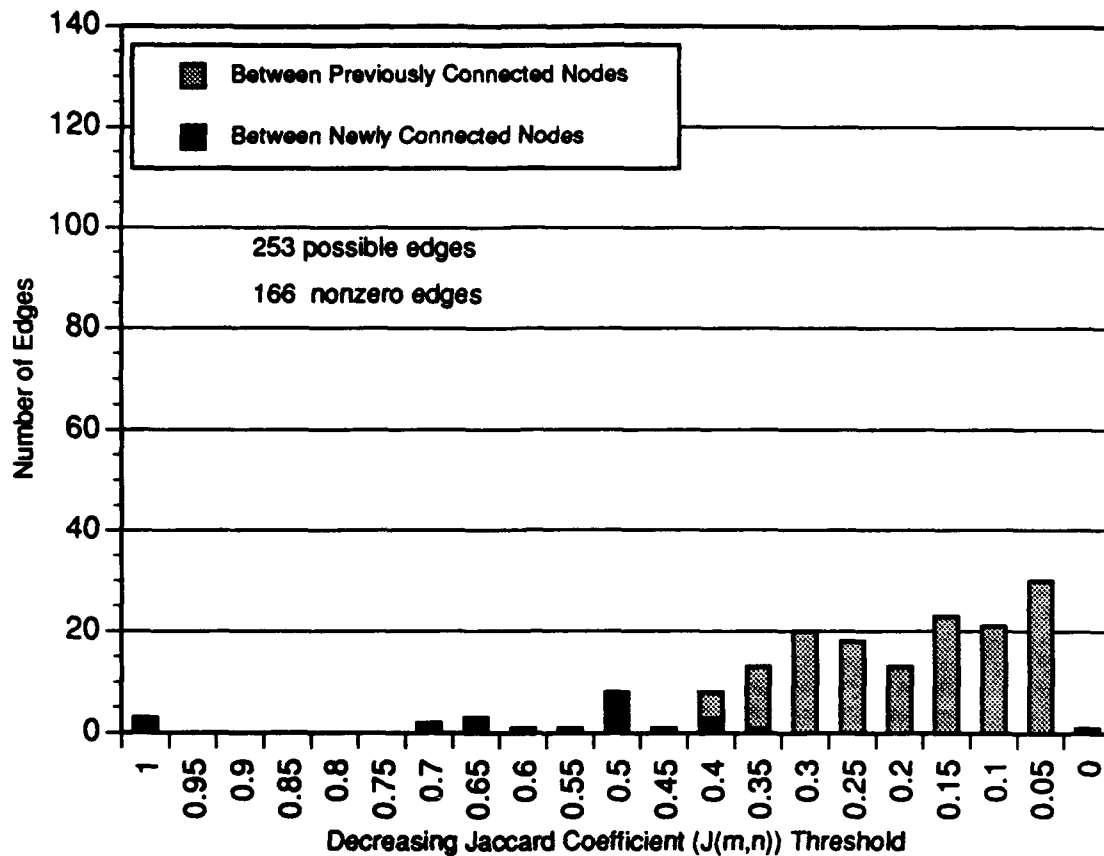


(a) Edges Added at Each Threshold

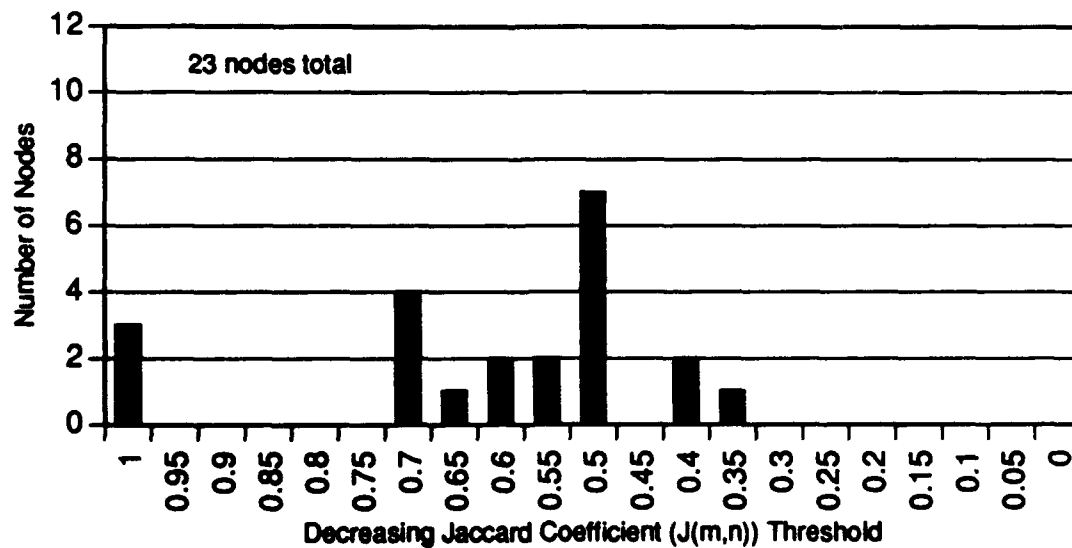


(b) Newly Connected Nodes at Each Threshold

Figure F.12 - Version R40, Coincidental Bounds

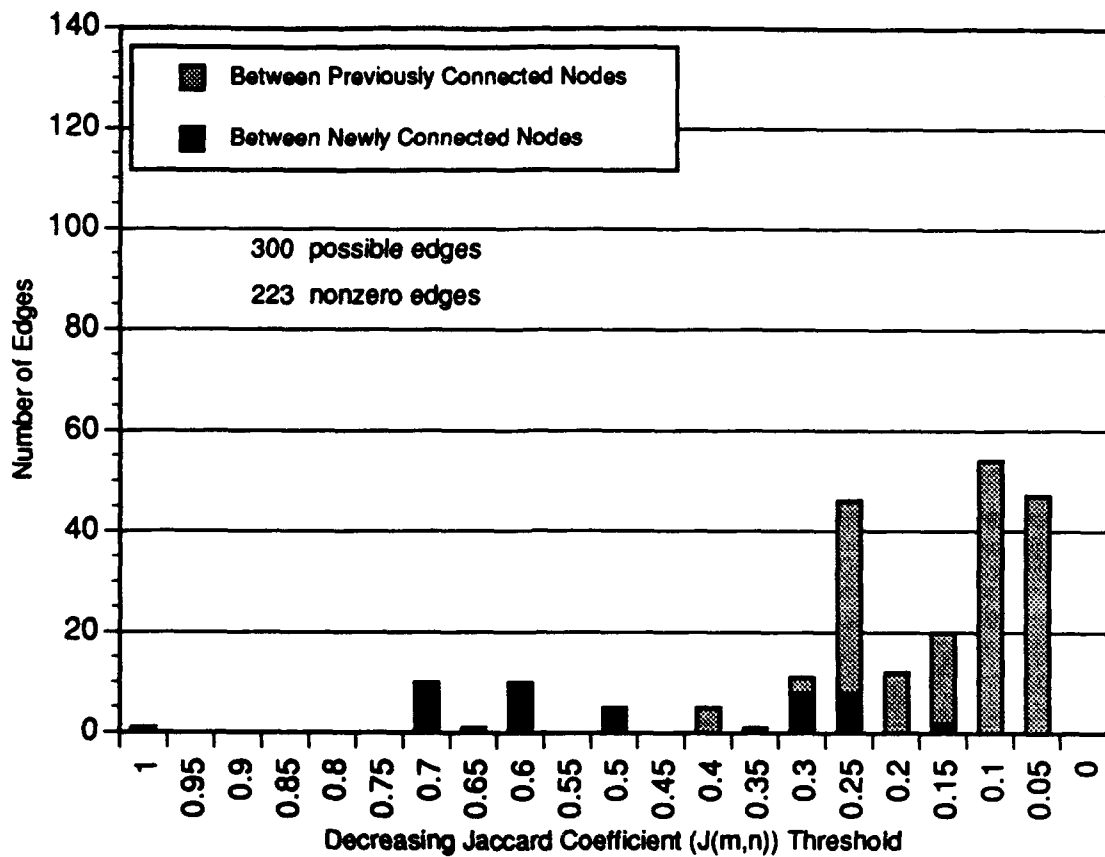


(a) Edges Added at Each Threshold

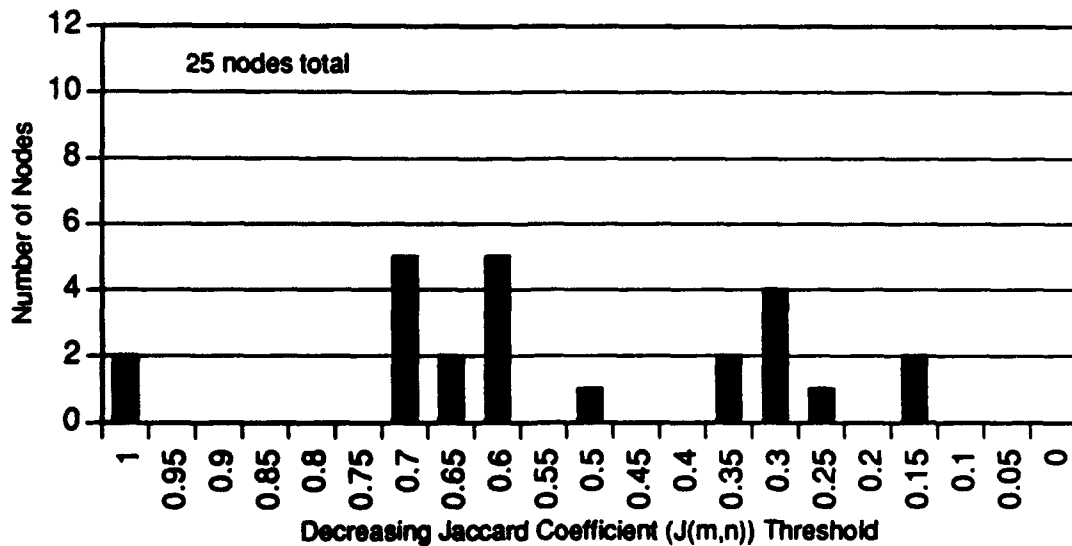


(b) Newly Connected Nodes at Each Threshold

Figure F.13 - Version 1, Composite Bounds

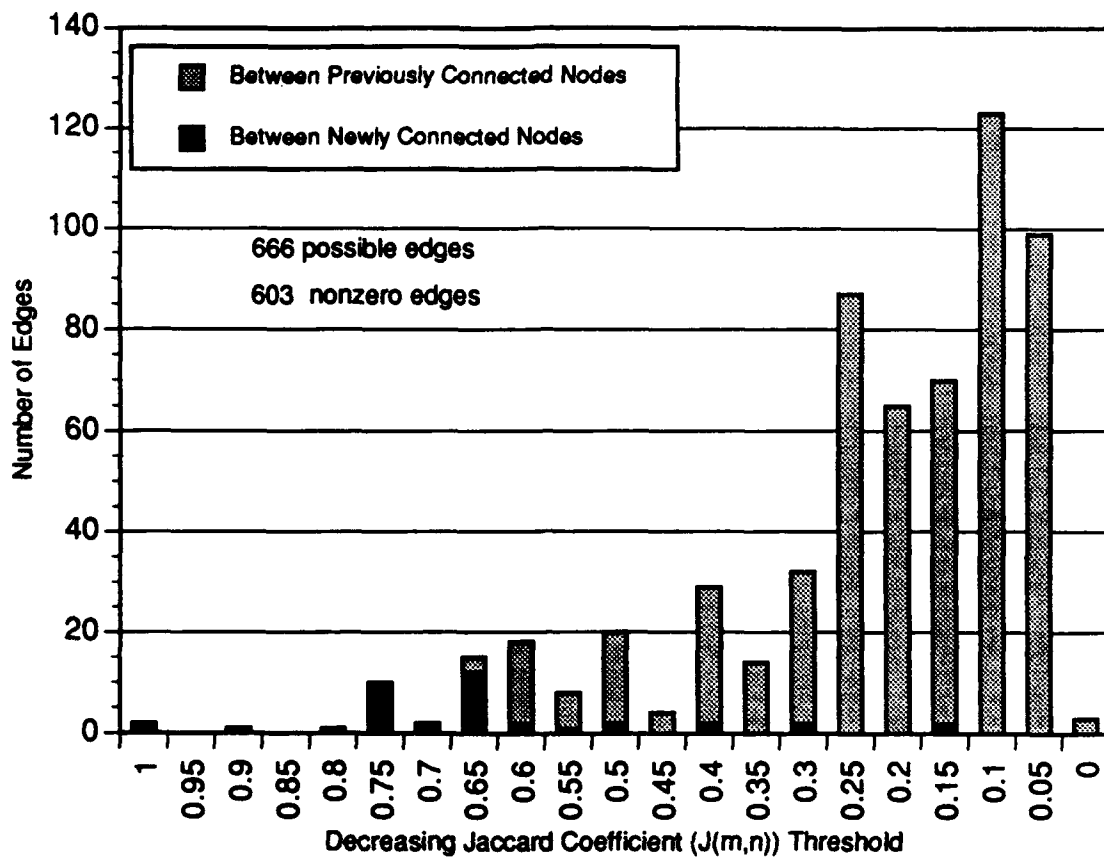


(a) Edges Added at Each Threshold

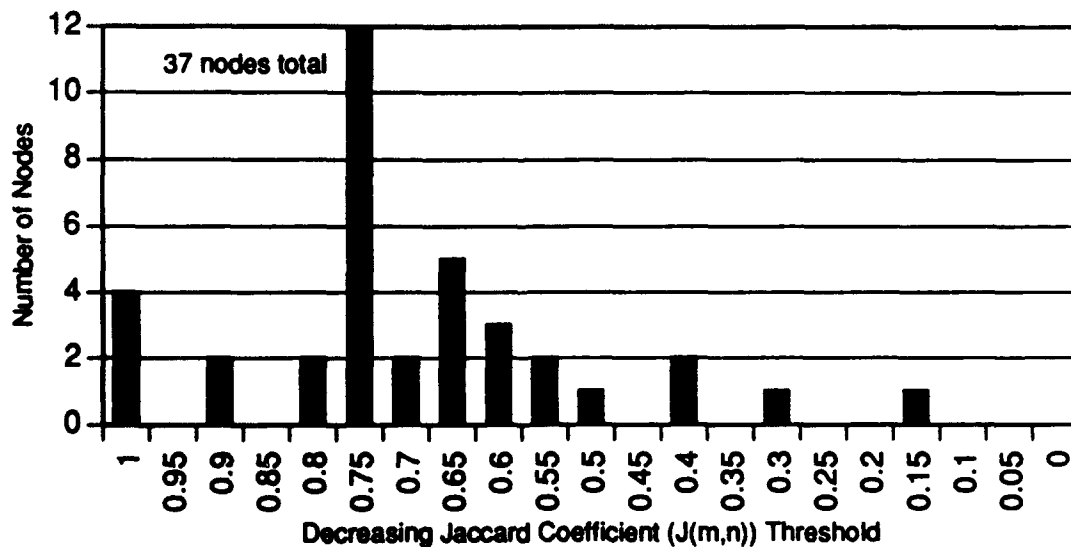


(b) Newly Connected Nodes at Each Threshold

Figure F.14 - Version 2, Composite Bounds

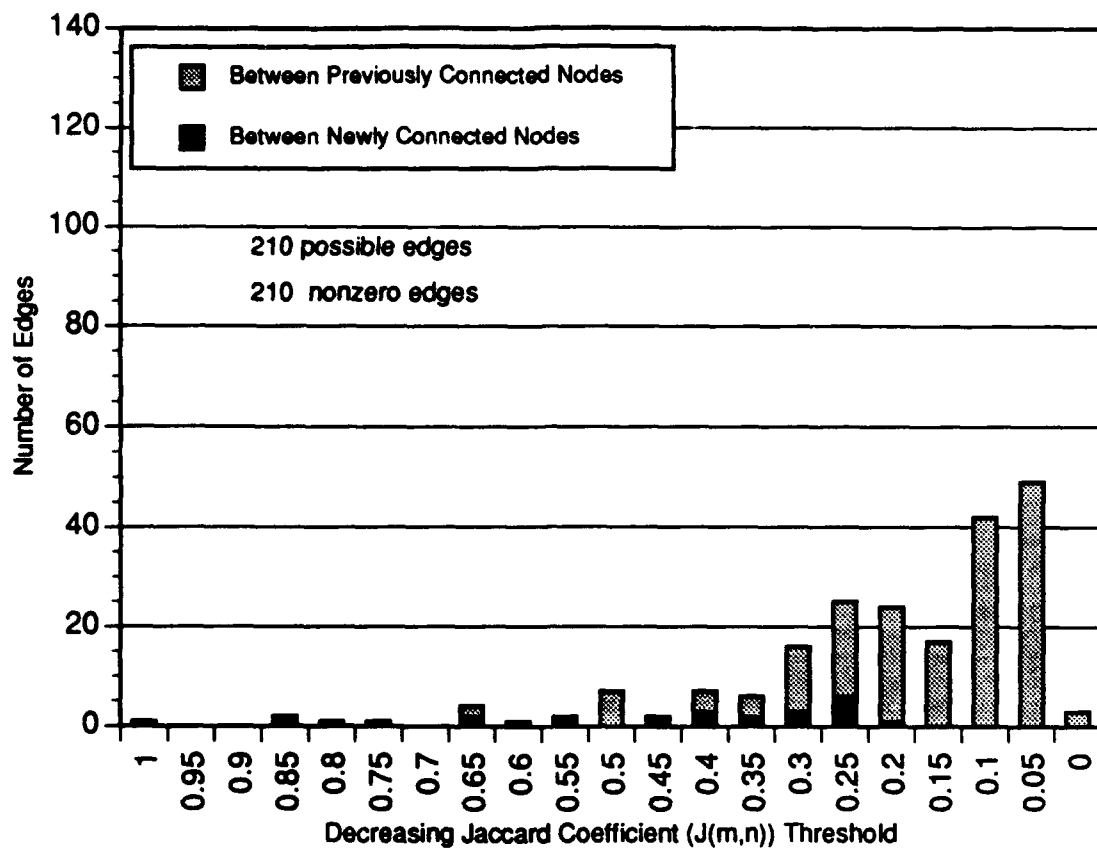


(a) Edges Added at Each Threshold

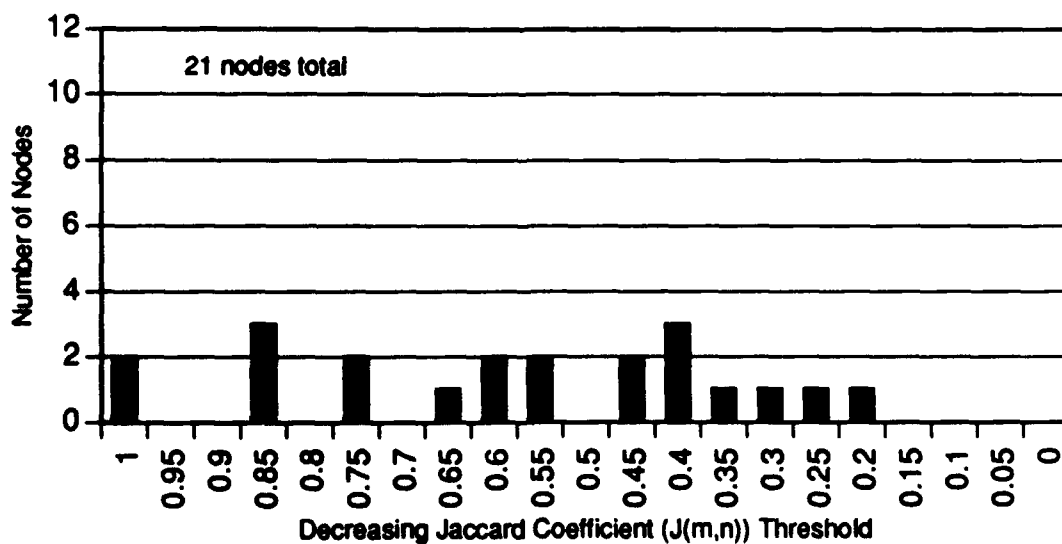


(b) Newly Connected Nodes at Each Threshold

Figure F.15 - Version 3, Composite Bounds

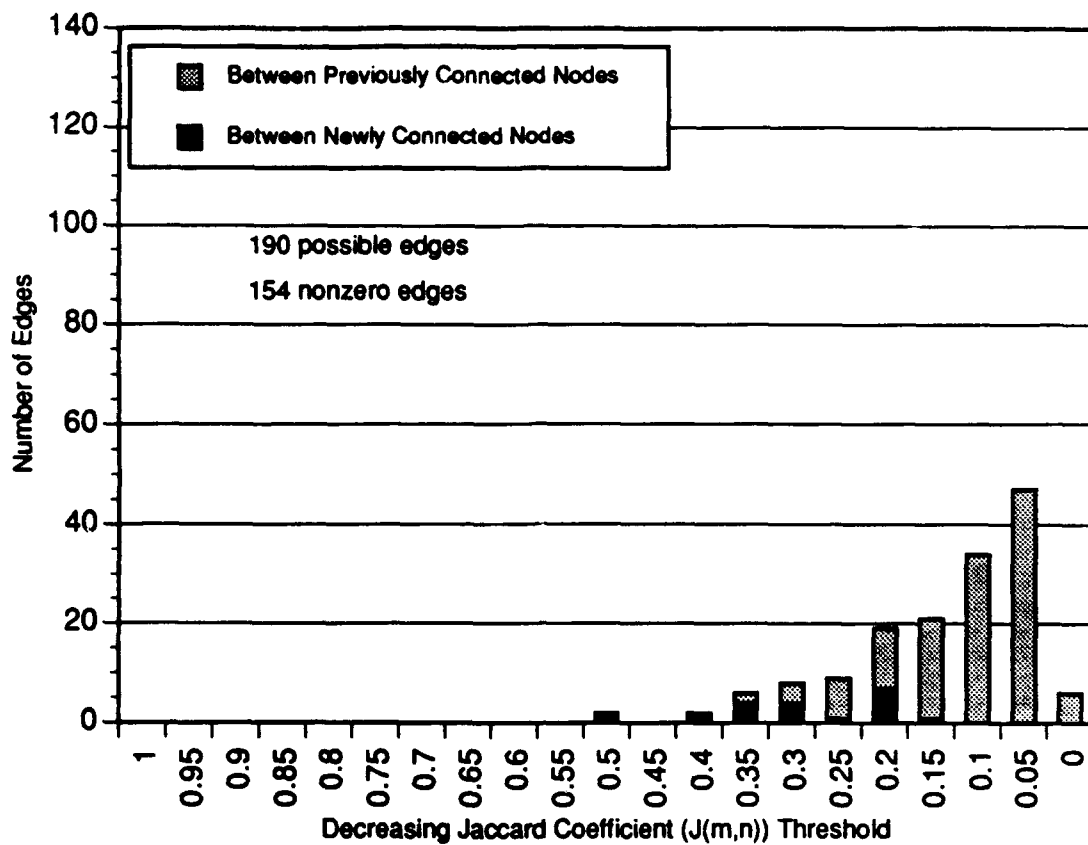


(a) Edges Added at Each Threshold

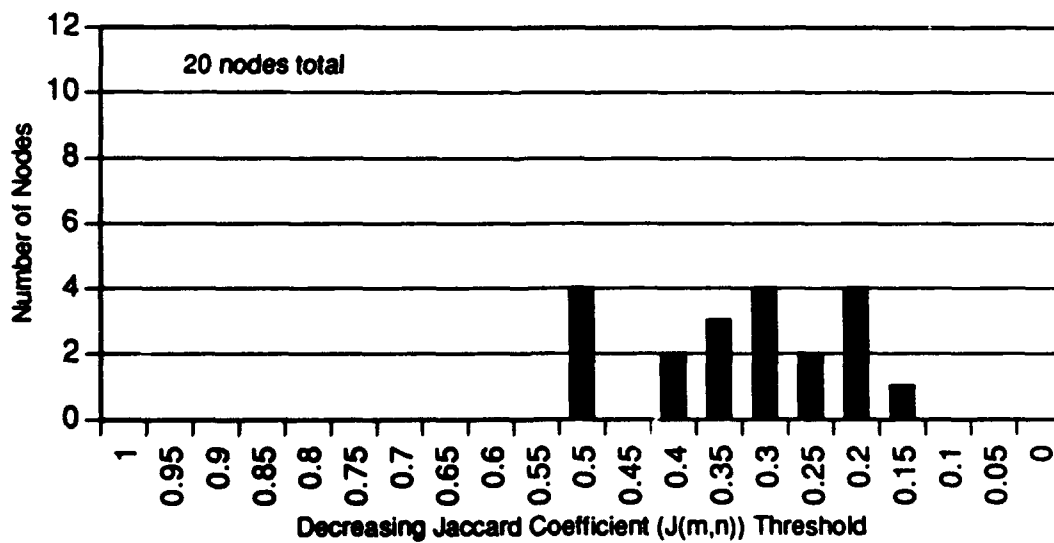


(b) Newly Connected Nodes at Each Threshold

Figure F.16 - Version 4, Composite Bounds

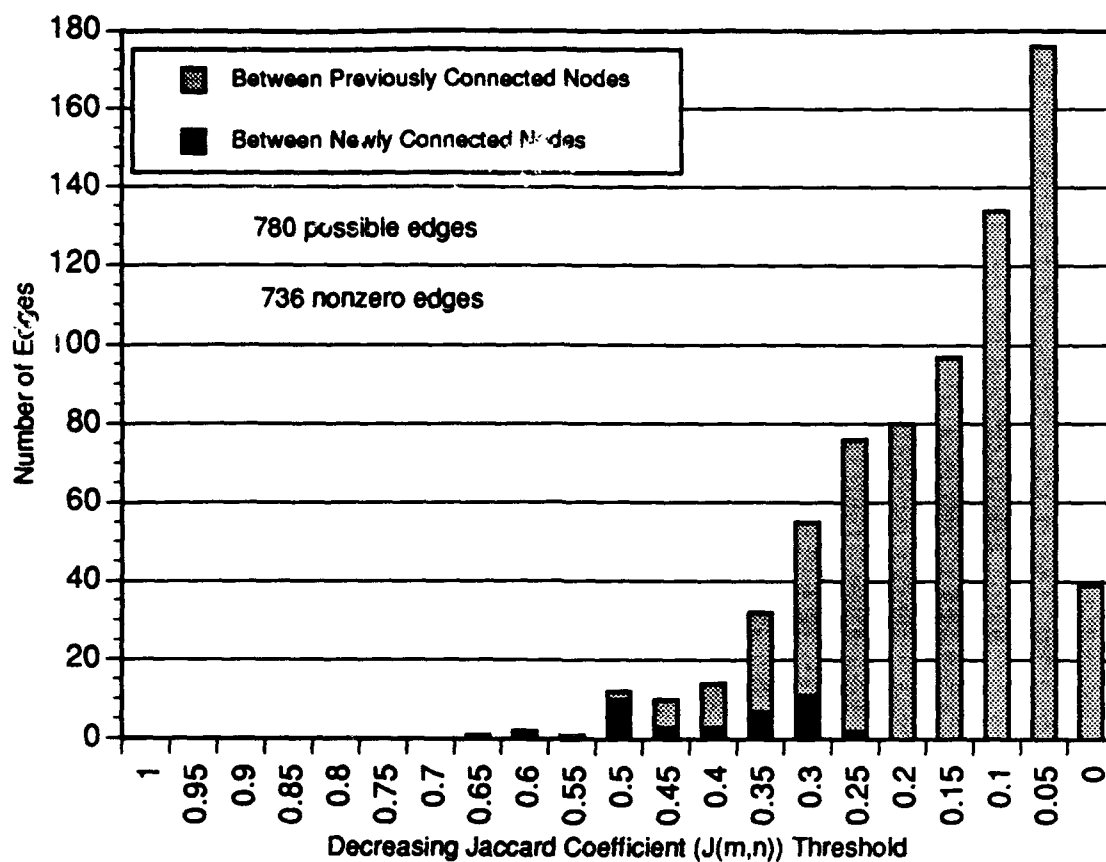


(a) Edges Added at Each Threshold

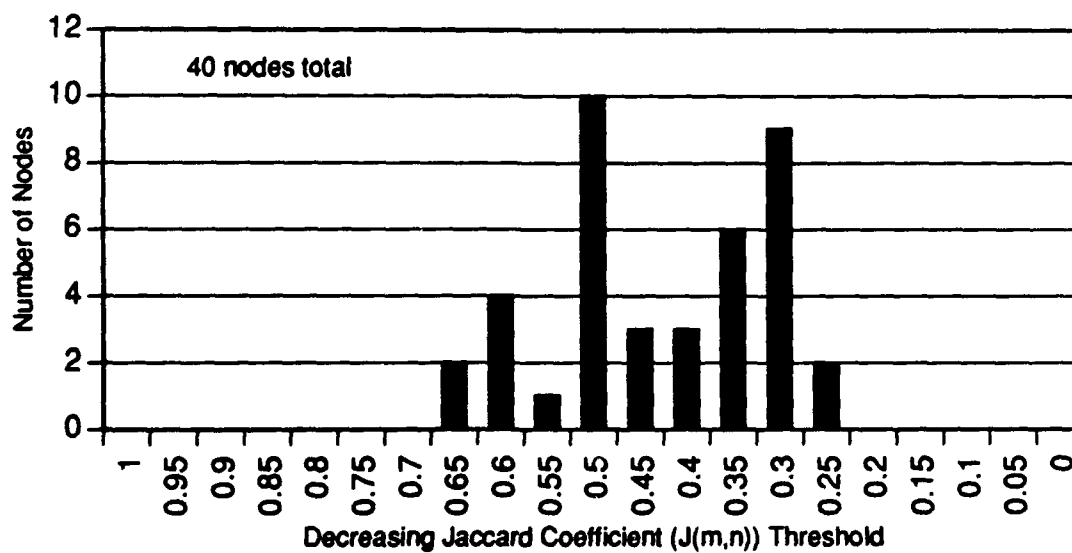


(b) Newly Connected Nodes at Each Threshold

Figure F.17 - Version R20, Composite Bounds



(a) Edges Added at Each Threshold



(b) Newly Connected Nodes at Each Threshold

Figure F.18 - Version R40, Composite Bounds

## LIST OF REFERENCES

- Alberts, D. S., "The Economics of Software Quality Assurance," Proceedings of the 1976 National Computer Conference, pp. 230-238, New York, New York, AFIPS Press, 1976.
- Ammann, Paul E., and Knight, John C., "Data Diversity: An Approach to Software Fault Tolerance," *IEEE Transactions on Computers*, pp. 418-425, April 1988.
- Beizer, Boris, *Software Testing Techniques*, Van Nostrand Reinhold, New York, New York, 1990.
- Bolchoz, John M., *The Identification of Software Failure Regions*, Master's Thesis, Naval Postgraduate School, Monterey, California, June 1990.
- Buckley, Fred, and Harary, Frank, *Distance in Graphs*, Addison-Wesley Publishing Company, Redwood City, California, 1990.
- DeMillo, Richard A., Lipton, Richard J., and Sayward, Frederick G., "Hints on Test Data Selection: Help for the Practicing Programmer," *Computer*, v. 11, pp. 34-41 April 1978.
- Godehardt, Erhard, *Graphs as Structural Models*, Advances in System Analysis, Volume 4, D. P. F. Moller, Editor, Friedr. Vieweg & Sohn, Braunschweig, West Germany, 1988.
- Goodenough, John B., and Gerhart, Susan L., "Toward a Theory of Test Data Selection," *IEEE Transactions on Software Engineering*, v. SE -1, no. 6, pp. 156-173, June 1975.
- Glossary of Software Engineering Terminology*, ANSI-IEEE STD 729-1983, Institute of Electrical and Electronics Engineers, 1983.
- Hamlet, Dick and Taylor, Ross, "Partition Testing Does Not Inspire Confidence," Second Workshop on Software Testing, Verification, and Analysis, Banff, Alberta, Canada, July 1988, pp. 206-216.
- Jain, Anil K., and Dubes, Richard C., *Algorithms for Clustering Data*, Prentice Hall, Englewood Cliffs, New Jersey, 1988.
- Myers, Glenford J., *The Art of Software Testing*, John Wiley and Sons, New York, New York, 1979.



Morell, Larry J. "A Theory of Fault-Based Testing," *IEEE Transactions on Software Engineering*, v. SE-16, no. 8, pp. 844-857, August 1990.

Offutt, A. Jefferson, "The Coupling Effect: Fact or Fiction?" Proceedings of the ACM SIGSOFT '89 Third Symposium on Software Testing, Analysis, and Verification, Key West, Florida, pp. 131-140, December, 1989.

Perlman, Gary, "The UNIX|STAT Handbook: Data Analysis Programs on UNIX and MSDOS," Wang Institute, Tyngsboro, Massachusetts, 1986.

Richardson, Debra J., and Clarke, Lori A., "A Partition Analysis Method to Increase Program Reliability," Proceedings of the Fifth International Conference on Software Engineering, 1981, pp. 244-253.

Richardson, Debra J., and Thompson, Margaret C., "The RELAY Model of Error Detection and its Application," Second Workshop on Software Testing, Verification, and Analysis, Banff, Alberta, Canada, pp. 223-230, July 1988.

Shimeall, Timothy J., *A Library of Failure Regions*, Naval Postgraduate School Technical Report NPSCS-91-002, September 1991.

Shimeall, Timothy J., Bolchoz, John M., and Griffin, Rachel, *Analytical Derivation of Software Failure Regions*, Naval Postgraduate School Technical Report NPS CS-91-002, 1991.

Shimeall, Timothy J., and Leveson, Nancy G., "An Empirical Comparison of Software Fault Tolerance and Fault Elimination," *IEEE Transactions on Software Engineering*, v. SE-17, no. 2, pp. 173-182, February 1991.

Voas, Jeffrey M. and Morell, Larry J., *Fault Sensitivity Analysis (PIA) Applied to Computer Programs*, College of William and Mary Technical Report WM-89-4, December 1989.

Wall Street Journal, "DSC Communications Says Modification of Software Led to Phone Disruptions," p. B3, 10 July 1991.

Weyuker, Elaine J. and Ostrand, Thomas J., "Theories of Program Testing and the Application of Revealing Subdomains," *IEEE Transactions on Software Engineering*, SE -6, no. 5, pp. 236-246, May 1980.

Weyuker, Elaine J. and Jeng, Bingchiang, "Analyzing Partition Testing Strategies," *IEEE Transactions on Software Engineering*, SE-17, no. 7, pp. 703-711, July 1991.

## INITIAL DISTRIBUTION LIST

Defense Technical Information Center Cameron Station Alexandria, VA 22304-6145	2
Dudley Knox Library Code 52 Naval Postgraduate School Monterey, CA 93943	2
Chairman, Code CS Naval Postgraduate School Monterey, CA 93943	2
Timothy J. Shimeall Code CS/Sm Naval Postgraduate School Monterey, CA 93943	6
LT Lelon Ginn 1008 Maple Dimmitt, TX 79027	2
CDR Rachel Griffin Naval ROTC University of Rochester Rochester, NY 14627	1